


11-24-00
EL666125529US

A

Please type a plus sign (+) inside this box 

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PTO/SB/05 (08-00)
Approved for use through 10/31/2002. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket No. PA1630US
First Inventor Mark Moriconi
Title System and Method for Maintaining Security
in a Distributed Computer Network
Express Mail Label No. EL666125529US

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

1. ☒ Fee Transmittal Form (e.g., PTO/SB/17)
(Submit an original and a duplicate for fee processing)
2. ☒ Applicant claims small entity status.
See 37 CFR 1.27.
3. ☒ Specification [Total Pages 51]
(preferred arrangement set forth below)
 - Descriptive title of the invention
 - Cross Reference to Related Applications
 - Statement Regarding Fed sponsored R & D
 - Reference to sequence listing, a table, or a computer program listing appendix
 - Background of the Invention
 - Brief Summary of the Invention
 - Brief Description of the Drawings (if filed)
 - Detailed Description
 - Claim(s)
 - Abstract of the Disclosure
4. ☒ Drawing(s) (35 U.S.C. 113) [Total Sheets 14]
5. Oath or Declaration [Total Pages 3]
 - a. ☐ Newly executed (original or copy)
 - b. ☒ Copy from a prior application (37 CFR 1.63 (d))
(for continuation/divisional with Box 17 completed)
 - i. ☐ **DELETION OF INVENTOR(S)**
Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).
6. ☐ Application Data Sheet. See 37 CFR 1.76

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

7. ☐ CD-ROM or CD-R in duplicate, large table or Computer Program (Appendix)
8. Nucleotide and/or Amino Acid Sequence Submission
(if applicable, all necessary)
 - a. ☐ Computer Readable Form (CRF)
 - b. Specification Sequence Listing on:
 - i. ☐ CD-ROM or CD-R (2 copies); or
 - ii. ☐ paper
 - c. ☐ Statements verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

9. ☐ Assignment Papers (cover sheet & document(s))
10. ☐ 37 CFR 3.73(b) Statement (when there is an assignee) ☐ Power of Attorney
11. ☐ English Translation Document (if applicable)
12. ☐ Information Disclosure Statement (IDS)/PTO-1449 ☐ Copies of IDS Citations
13. ☒ Preliminary Amendment
14. ☒ Return Receipt Postcard (MPEP 503)
(Should be specifically itemized)
15. ☐ Certified Copy of Priority Document(s)
(if foreign priority is claimed)
16. ☐ Other:

17. If a CONTINUING APPLICATION, check appropriate box, and supply the requisite information below and in a preliminary amendment, or in an Application Data Sheet under 37 CFR 1.76:

☒ Continuation ☐ Divisional ☐ Continuation-in-part (CIP)

of prior application No: 09 / 248,788

Prior application information.

Examiner Willie R. Martin

Group I Art Unit. 2787

For CONTINUATION OR DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 5b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

18. CORRESPONDENCE ADDRESS

☒ Customer Number or Bar Code Label



or ☐ Correspondence address below

Name

22830

PATENT TRADEMARK OFFICE

Address

City

State

Zip Code

Country

Telephone

Fax

Name (Print/Type)

David Lewis

Registration No. (Attorney/Agent)

33,101

Signature

David Lewis

Date November 22, 2000

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

FEE TRANSMITTAL for FY 2001

Patent fees are subject to annual revision.

TOTAL AMOUNT OF PAYMENT (\$1,770.00)

Complete if Known

| | |
|----------------------|-------------------|
| Application Number | Unknown |
| Filing Date | November 22, 2000 |
| First Named Inventor | Mark Moriconi |
| Examiner Name | Unknown |
| Group Art Unit | Unknown |
| Attorney Docket No. | PA1630US |

METHOD OF PAYMENT

1. ☒ The Commissioner is hereby authorized to charge indicated fees and credit any overpayments to:

Deposit Account Number 06-0600

Deposit Account Name Carr & Ferrell, LLP

☒ Charge Any Additional Fee Required Under 37 CFR 1.16 and 1.17

☒ Applicant claims small entity status. See 37 CFR 1.27

2. ☒ Payment Enclosed:

☒ Check ☐ Credit card ☐ Money Order ☐ Other

FEE CALCULATION

1. BASIC FILING FEE

| Large Entity Code (\$) | Small Entity Code (\$) | Fee Description | Fee Paid |
|------------------------|------------------------|------------------------|----------|
| 101 710 | 201 355 | Utility filing fee | 355 |
| 106 320 | 206 160 | Design filing fee | |
| 107 490 | 207 245 | Plant filing fee | |
| 108 710 | 208 355 | Reissue filing fee | |
| 114 150 | 214 75 | Provisional filing fee | |

SUBTOTAL (1) (\$355)

2. EXTRA CLAIM FEES

| Total Claims | Extra Claims | Fee from below | Fee Paid | |
|--------------------|--------------|----------------|----------|-------|
| 35 | -20** = 15 | 9 | 135 | |
| Independent Claims | 35 | -3** = 32 | 40 | 1,280 |
| Multiple Dependent | | | | 0 |

| Large Entity Code (\$) | Small Entity Code (\$) | Fee Description |
|------------------------|------------------------|--|
| 103 18 | 203 9 | Claims in excess of 20 |
| 102 80 | 202 40 | Independent claims in excess of 3 |
| 104 270 | 204 135 | Multiple dependent claim, if not paid |
| 109 80 | 209 40 | ** Reissue independent claims over original patent |
| 110 18 | 210 9 | ** Reissue claims in excess of 20 and over original patent |

SUBTOTAL (2) (\$1,415)

**for number previously paid, if greater; For Reissues, see above

FEE CALCULATION (continued)

3. ADDITIONAL FEES

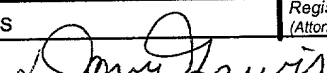
| Large Entity Code (\$) | Small Entity Code (\$) | Fee Description | Fee Paid |
|------------------------|------------------------|--|----------|
| 105 130 | 205 65 | Surcharge - late filing fee or oath | |
| 127 50 | 227 25 | Surcharge - late provisional filing fee or cover sheet | |
| 139 130 | 139 130 | Non-English specification | |
| 147 2,520 | 147 2,520 | For filing a request for ex parte reexamination | |
| 112 920* | 112 920* | Requesting publication of SIR prior to Examiner action | |
| 113 1,840* | 113 1,840* | Requesting publication of SIR after Examiner action | |
| 115 110 | 215 55 | Extension for reply within first month | |
| 116 390 | 216 195 | Extension for reply within second month | |
| 117 890 | 217 445 | Extension for reply within third month | |
| 118 1,390 | 218 695 | Extension for reply within fourth month | |
| 128 1,890 | 228 945 | Extension for reply within fifth month | |
| 119 310 | 219 155 | Notice of Appeal | |
| 120 310 | 220 155 | Filing a brief in support of an appeal | |
| 121 270 | 221 135 | Request for oral hearing | |
| 138 1,510 | 138 1,510 | Petition to institute a public use proceeding | |
| 140 110 | 240 55 | Petition to revive - unavoidable | |
| 141 1,240 | 241 620 | Petition to revive - unintentional | |
| 142 1,240 | 242 620 | Utility issue fee (or reissue) | |
| 143 440 | 243 220 | Design issue fee | |
| 144 600 | 244 300 | Plant issue fee | |
| 122 130 | 122 130 | Petitions to the Commissioner | |
| 123 50 | 123 50 | Petitions related to provisional applications | |
| 126 240 | 126 240 | Submission of Information Disclosure Stmt | |
| 581 40 | 581 40 | Recording each patent assignment per property (times number of properties) | |
| 146 710 | 246 355 | Filing a submission after final rejection (37 CFR § 1.129(a)) | |
| 149 710 | 249 355 | For each additional invention to be examined (37 CFR § 1.129(b)) | |
| 179 710 | 279 355 | Request for Continued Examination (RCE) | |
| 169 900 | 169 900 | Request for expedited examination of a design application | |

Other fee (specify) _____

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) (\$0)

SUBMITTED BY

| | | | | | |
|-------------------|---|-----------------------------------|-------------------|-----------|----------------|
| Name (Print/Type) | David Lewis | Registration No. (Attorney/Agent) | 33,101 | Telephone | (650) 812-3412 |
| Signature |  | Date | November 22, 2000 | | |

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

**IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE**

APPLICANT: Mark Moriconi, et al.
SERIAL NO.: Unknown
S.N. in parent: 09/248,788
FILING DATE: Herewith
TITLE: System and Method for Maintaining Security in a Distributed
Computer Network
EXAMINER: Unknown
Examiner in parent: W. Martin
ART UNIT: 2787
ATTY. DKT. NO: PA1630US

COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

PRELIMINARY AMENDMENT

In response to the office action in the parent Application S.N. 09/248,788, mailed
December 30, 1999 (paper #3) please amend the specification as follows:

In the specification

On page 1, line 5 after “application is” insert - - a continuation of U.S. Patent Application Serial Number 09/248,788, entitled, “SYSTEM AND METHOD FOR MAINTAINING SECURITY IN A DISTRIBUTED COMPUTER NETWORK,” filed February 12, 1999, which is- -.

Delete from page 8, line 3 through page 9, line 9.

On page 21, line 4 after the word “subject,” replace the word “math” with the word - - match - -.

In the claims

Cancel claims 1-15, 17- 28, and 30-56 without prejudice or disclaimer.

1 16. (Once Amended) A system for maintaining security in a distributed computing
2 environment, comprising:
3 a policy manager for managing a security policy; and
4 an application guard for managing access to securable components as specified by
5 the security policy;
6 [The system of claim 1,] wherein the application guard further allows for
7 additional customized code to process and evaluate authorization requests
8 based on the additional customized code.

1 29. (Once Amended) A system for controlling user access in a distributed computing
2 environment, comprising:
3 a global policy specifying access privileges of the user to securable components;
4 a policy manager located on a server for managing and distributing a local client
5 policy based on the global policy to a client, and
6 an application guard located on the client for managing access to the securable
7 components as specified by the local client policy;
8 [The system of claim 18,] wherein the application guard further allows for
9 additional customized code to process and evaluate authorization requests
10 based on the additional customized code.

Add new claims 57-89

1 57. A system for maintaining security in a distributed computing environment,
2 comprising:
3 a policy manager for managing a security policy; and
4 an application guard for managing access to securable components including at
5 least one application as specified by the security policy.

1 58. A system for maintaining security in a distributed computing environment,
2 comprising:
3 a policy manager for managing a security policy; and
4 an application guard for managing access to securable components including a
5 function within an application as specified by the security policy.

1 59. A system for maintaining security in a distributed computing environment,
2 comprising:
3 a policy manager for managing a security policy; and
4 an application guard for managing access to securable components including a
5 procedure within an application as specified by the security policy.

1 60. A system for maintaining security in a distributed computing environment,
2 comprising:
3 a policy manager for managing a security policy; and
4 an application guard for managing access to securable components including a
5 data structure within an application as specified by the security policy.

1 61. A system for maintaining security in a distributed computing environment,
2 comprising:
3 a policy manager for managing a security policy; and
4 an application guard for managing access to securable components including a
5 database object referenced by an application as specified by the security
6 policy.

1 62. A system for maintaining security in a distributed computing environment,
2 comprising:
3 a policy manager for managing a security policy; and
4 an application guard for managing access to securable components including a
5 file system object referenced by an application as specified by the security
6 policy.

1

1 63. A method for maintaining security in a distributed computing environment,
2 comprising:
3 managing a security policy via a policy manager; and
4 managing access via an application guard to securable components including at
5 least one application as specified by the security policy.

1 64. A method for maintaining security in a distributed computing environment,
2 comprising:
3 managing a security policy via a policy manager; and
4 managing access via an application guard to securable components including a
5 function within an application as specified by the security policy.

1 65. A method for maintaining security in a distributed computing environment,
2 comprising:
3 managing a security policy via a policy manager; and
4 managing access via an application guard to securable components including a
5 procedure within an application as specified by the security policy.

1 66. A method for maintaining security in a distributed computing environment,
2 comprising:
3 managing a security policy via a policy manager; and
4 managing access via an application guard to securable components including a
5 data structure within an application as specified by the security policy.

1 67. A method for maintaining security in a distributed computing environment,
2 comprising:
3 managing a security policy via a policy manager; and
4 managing access via an application guard to securable components including a
5 database object referenced by an application as specified by the security
6 policy.

1 68. A method for maintaining security in a distributed computing environment,
2 comprising:
3 managing a security policy via a policy manager; and
4 managing access via an application guard to securable components including a
5 file system object referenced by an application as specified by the security
6 policy.

1 69. A method for maintaining security in a distributed computing environment,
2 comprising:
3 managing a security policy via a policy manager; and
4 managing via an application guard access to securable components as specified by
5 the security policy;
6 wherein the application guard further allows for additional customized code to
7 process and evaluate authorization requests based on the additional
8 customized code.

1 70. A method for controlling user access via a system in a distributed computing
2 environment, comprising:
3 specifying access privileges of the user via a global policy to securable
4 components;
5 managing and distributing via a policy manager a local client policy based on the
6 global policy located on a server to a client, and
7 managing access via an application guard located on the client to the securable
8 components as specified by the local client policy;
9 wherein the application guard further allows for additional customized code to
10 process and evaluate authorization requests based on the additional
11 customized code.

1 71. A method for authorization that provides access to securable components of a
2 system for a user, comprising:
3 specifying access privileges of the user via a policy to the securable components
4 managing access via an application guard to the securable components; and
5 executing via a processor coupled to said system said application guard;
6 wherein the application guard further allows for additional customized code to
7 process and evaluate authorization requests based on the additional
8 customized code.

1 72. A method for providing a system for maintaining security in a distributed
2 computing environment, comprising:
3 providing a policy manager for managing a security policy; and
4 providing an application guard for managing access to securable components
5 including at least one application as specified by the security policy.

1 73. A method for providing a system for maintaining security in a distributed
2 computing environment, comprising:
3 providing a policy manager for managing a security policy; and
4 providing an application guard for managing access to securable components
5 including a function within an application as specified by the security
6 policy.

1 74. A method for providing a system for maintaining security in a distributed
2 computing environment, comprising:
3 providing a policy manager for managing a security policy; and
4 providing an application guard for managing access to securable components
5 including a procedure within an application as specified by the security
6 policy.

1 75. A method for providing a system for maintaining security in a distributed
2 computing environment, comprising:
3 providing a policy manager for managing a security policy; and
4 providing an application guard for managing access to securable components
5 including a data structure within an application as specified by the security
6 policy.

1 76. A method for providing a system for maintaining security in a distributed
2 computing environment, comprising:
3 providing a policy manager for managing a security policy; and
4 providing an application guard for managing access to securable components
5 including a database object referenced by an application as specified by
6 the security policy.

1 77. A method for providing a system for maintaining security in a distributed
2 computing environment, comprising:
3 providing a policy manager for managing a security policy; and
4 providing an application guard for managing access to securable components
5 including a file system object referenced by an application as specified by
6 the security policy.

1 78. A method for providing a system for maintaining security in a distributed
2 computing environment, comprising:
3 providing a policy manager for managing a security policy; and
4 providing an application guard for managing access to securable components as
5 specified by the security policy;
6 wherein the application guard further allows for additional customized code to
7 process and evaluate authorization requests based on the additional
8 customized code.

1 79. A method for providing a system for controlling user access in a distributed
2 computing environment, comprising:
3 providing a global policy specifying access privileges of the user to securable
4 components;
5 providing a policy manager located on a server for managing and distributing a
6 local client policy based on the global policy to a client, and
7 providing an application guard located on the client for managing access to the
8 securable components as specified by the local client policy;
9 wherein the application guard further allows for additional customized code to
10 process and evaluate authorization requests based on the additional
11 customized code.

1 80. A method for providing a system for authorization that provides access to
2 securable components for a user, comprising:
3 providing a policy specifying access privileges of the user to the securable
4 components;
5 providing an application guard; and
6 providing a processor coupled to said system, said processor executing said
7 application guard to manage access to the securable components;
8 wherein the application guard further allows for additional customized code to
9 process and evaluate authorization requests based on the additional
10 customized code.

1 81. A computer readable storage medium having stored thereon a method for
2 maintaining security in a distributed computing environment comprising the steps of:
3 managing a security policy via a policy manager; and
4 managing access via an application guard to securable components including at
5 least one application as specified by the security policy.

1 82. A computer readable storage medium having stored thereon a method for
2 maintaining security in a distributed computing environment comprising the steps of:
3 managing a security policy via a policy manager; and
4 managing access via an application guard to securable components including a
5 function within an application as specified by the security policy.

1 83. A computer readable storage medium having stored thereon a method for
2 maintaining security in a distributed computing environment comprising the steps of:
3 managing a security policy via a policy manager; and
4 managing access via an application guard to securable components including a
5 procedure within an application as specified by the security policy.

1 84. A computer readable storage medium having stored thereon a method for
2 maintaining security in a distributed computing environment comprising the steps of:
3 managing a security policy via a policy manager; and
4 managing access via an application guard to securable components including a
5 data structure within an application as specified by the security policy.

1 85. A computer readable storage medium having stored thereon a method for
2 maintaining security in a distributed computing environment comprising the steps of:
3 managing a security policy via a policy manager; and
4 managing access via an application guard to securable components including a
5 database object referenced by an application as specified by the security
6 policy.

1 86. A computer readable storage medium having stored thereon a method for
2 maintaining security in a distributed computing environment, comprising:
3 managing a security policy via a policy manager; and
4 managing access via an application guard to securable components including a
5 file system object referenced by an application as specified by the security
6 policy.

1 87. A computer readable storage medium having stored thereon a method for
2 maintaining security in a distributed computing environment comprising the steps of:
3 managing a security policy via a policy manager; and
4 managing via an application guard access to securable components as specified by
5 the security policy;
6 wherein the application guard further allows for additional customized code to
7 process and evaluate authorization requests based on the additional
8 customized code.

1 88. A computer readable storage medium having stored thereon a method for
2 controlling user access via a system in a distributed computing environment, comprising
3 the steps of:
4 specifying access privileges of the user via a global policy to securable
5 components;
6 managing and distributing via a policy manager a local client policy based on the
7 global policy located on a server to a client, and
8 managing access via an application guard located on the client to the securable
9 components as specified by the local client policy;
10 wherein the application guard further allows for additional customized code to
11 process and evaluate authorization requests based on the additional
12 customized code.

1 89. A computer readable storage medium having stored thereon a method for
2 authorization that provides access to securable components of a system for a user,
3 comprising:
4 specifying access privileges of the user via a policy to the securable components
5 managing access via an application guard to the securable components; and
6 executing via a processor coupled to said system said application guard;
7 wherein the application guard further allows for additional customized code to
8 process and evaluate authorization requests based on the additional
9 customized code.

Remarks

Claims 16 and 29 of the parent Application were indicated as containing allowable subject matter but were objected to as dependent upon a rejected claim. Accordingly claims 16 and 29 are being placed into independent form and are therefore allowable.

Claims 69, 78 and 87 are the same as claim 16 except claim 69 is worded as a method for maintaining security, claim 78 is worded as a method of providing, and claim 87 is a computer readable storage medium storing the method of claim 69. Similarly, claims 70, 79, and 88 are the same as claim 29 except claim 70 is worded as a method for controlling user access, claim 79 is worded as a method of providing, and claim 88 is a computer readable storage medium storing the method of claim 68. Also claims 71, 80 and 89 are the same as allowed claim 31 of the parent Application except claim 71 is worded as a method for authorization, claim 80 is worded as a method of providing, and claim 89 is a computer readable storage medium storing the method of claim 70. Therefore claims 69-71, 78-80 and 87-89 contain all of the limitations of allowable claims 16 and 29 and allowed claim 31 of the parent Application and are allowable.

Each of claims 57-62 corresponds to a different element from the Markush group of allowed claim 1 of the parent Application. Otherwise claims 57-62 are identical to allowed claim 1 of the parent Application and are therefore allowable.

Claims 63-68, 72-77 and 81-86 are the same as claims 57-62, respectively, except claims 63-67 are worded as methods of maintaining security, claims 72-77 are worded as methods of providing, and claims 81-86 are computer readable storage mediums storing the methods of claim 63-68, respectively. Therefore claims 63-68, 72-77 and 81-86

contain all of the limitations of allowed claim 1 of the parent Application and therefore are allowable.

Therefore the Examiner is respectfully requested to allow the claims 16, 29, and 57-89. If the Examiner has questions regarding this case, the Examiner is invited to contact Applicant's undersigned representative at the telephone number listed below.

Respectfully submitted,

Mark Moriconi, et al.

Date: November 22, 2000

By:



David Lewis, Reg. No. 33,101
Carr & Ferrell *LLP*
2225 East Bayshore Road, Suite 200
Palo Alto, CA 94303
Phone (650) 812-3412
Fax (650) 812-3444

SYSTEM AND METHOD FOR MAINTAINING SECURITY

IN A DISTRIBUTED COMPUTER NETWORK

CROSS-REFERENCE TO RELATED APPLICATIONS

5 The present application is related to, and claim priority in, co-
pending U.S. Provisional Patent Application Serial No. 60/105,963,
entitled "System And Method For Maintaining Security In A Distributed
Computer Network," filed on October 28, 1998.

BACKGROUND OF THE INVENTION

1. Field of the Invention

10 This invention relates generally to computer security systems, and
15 relates more particularly to a system and method for managing and
enforcing complex security requirements in a distributed computer
network.

2. Description of the Background Art

20 Computer security issues have become more complex with the
continual evolution of contemporary computer systems. As corporations
utilize increasingly distributed and open computing environments, the

security requirements of an enterprise typically grow accordingly. The complexity of employee, customer and partner access to critical information assets, while assuring proper security, has proven to be a major hurdle. For example, many organizations deploy applications that allow their external business partners, as well as their own internal employees, to access sensitive information resources within the enterprise. In the absence of adequate security measures, an enterprise may thus be subject to the risk of decreased security and confidentiality.

While most organizations focus their security concerns on protecting the internal network from the outside world, it is estimated that 80-90% of all corporate security breaches come from within an organization (source: Aberdeen Group, September 1997). This further underscores the need to specify and enforce an access control security policy within the enterprise network.

In today's complex business environment, specifying, stating, implementing and managing an enterprise access control policy may be both difficult and inefficient. When corporate data and applications revolved around a mainframe model, the problem of defining and managing access to corporate applications was relatively straightforward. Today, the complexity of business methods, as well as the complexity of distributed application architectures, may force companies to resort to manual, ineffective or highly custom approaches to access control in their attempts to implement the business process.

To secure a complex and distributed computer system, the system may typically employ a combination of encryption, authentication, and authorization technologies. Encryption is a means of sending information between participants in a manner that prevents other parties
5 from reading the information. Authentication is a process of verifying a party's identity. Authorization is a technique for determining what actions a participant is allowed to perform.

Encryption and authentication are well-understood and have led to effective network security products, whereas authorization technology is
10 not as well developed, and is often inadequate for many enterprises. The security approach of most companies today is to focus on the authentication of users to ensure that those users are part of the organization or a member of a select group. Authentication can be accomplished with a number of different approaches, from simple
15 password or challenge response mechanisms to smart cards and biometric devices such as a fingerprint reader. Once users are authenticated, however, there is still a significant problem in managing and enforcing their set of privileges, which may be unique and vary widely between users. The same authentication mechanism can be used
20 for every user, but different authorization mechanisms must be developed for most applications. Therefore, reliable and efficient access control is a much more difficult problem facing enterprises today.

Authentication mechanisms often work together with some sort of access control facility that can protect information resources from unauthorized users. Examples of network security products include firewalls, digital certificates, virtual private networks, and single sign-on systems. Some of these products provide limited support for resource-level authorization. For example, a firewall can screen access requests to an application or a database, but does not provide object-level authorization within an application or database. Single Sign-On (SSO) products, for example, maintain a list of resources an authenticated user can access by managing the login process to many different applications. However, firewalls, SSO and other related products are very limited in their ability to implement a sophisticated security policy characteristic of many of today's enterprises. They are limited to attempting to manage access at a login, or "launch level", which is an all or nothing approach that inherently cannot implement a business-level policy.

A real-world security policy within a large enterprise is a detailed and dynamic knowledge base specific to that organization. The authorization privileges are specific to the constantly evolving set of users, applications, partners, and global policies that the enterprise puts in place to protect its key information resources. A security policy within a large enterprise can consist of tens or hundreds of thousands of individual rules that cover which users are authorized to access particular applications, perform various operations, or manage the

delegation and transfer of tasks. Many of these policy rules that implement the business practice of the organization have to be hard coded within custom-built applications or stored in the database.

The key problem is that these policy rules are localized, scattered throughout the organization, and embedded in applications and databases. Such embedding is expensive and error-prone, and mitigates against efficient policy updates. An organization cannot effectively implement and manage the resulting policy. Inconsistencies arise and updates can quickly become unmanageable. Policy queries and analysis from a global perspective are nearly impossible. The resulting policy begins to diverge from the intended business practices of the organization. Compromises are made in the policy implementation at the department level, and auditors can quickly become frustrated.

The increasing security risks associated with the proliferation of distributed computing, including Intranet and Extranet applications, are prompting many organizations to explore a broad range of security solutions for controlling access to their important information assets. Although organizations have a number of solutions to choose from for authenticating users (determining and verifying who is attempting to gain access to the network or individual applications), there is little choice when it comes to controlling what users can do and when they can do it to the extent necessary to implement the kinds of complex security policies required by modern organizations. Organizations have

been forced to choose between custom authorization solutions that are costly, error-prone, and difficult to manage, or third-party solutions that are very limited in their ability to control access to information across applications and databases.

5 A real-world security policy within a large organization is a detailed and dynamic knowledge base that determines which users are authorized to access particular applications, perform various operations or manage the delegation and transfer of tasks, as well as when and under what circumstances they are permitted to do so. Authorization
10 privileges depend upon a constantly evolving set of users, applications, partners, and business policies that comprise the enterprise security policy. A typical enterprise environment consists of several thousand users, hundreds of applications, and a myriad of network resources, resulting in a security policy that can consist of tens or hundreds of
15 thousands of interrelated policy rules.

 Typically, organizations attempt to control access to the internals of in-house applications through policy rules that are hard-coded in the application or through stored procedure statements in the database. But as the number of applications and databases grows, this patchwork
20 approach to authorization quickly gets out of hand. First, organizations must incur the costly and time-consuming overhead of developing customized security code for each application. But more importantly, once the code is developed and embedded in an application, the

embedded policy rules become impossible to track, difficult to update, and nearly impossible to manage because they are scattered throughout the organization.

With an estimated 80 percent of all security breaches coming from authorized users (source: Forrester Research), advanced policy features and enforcement mechanisms are needed to control access to sensitive information assets. To implement an enterprise policy, organizations need a centralized policy and a powerful way to specify policy rules to give them adequate access control security. At the same time, they need a distributed authorization infrastructure to provide authorization services to all applications with performance and scalability for modern distributed network environments.

Therefore, for the foregoing reasons, an improved system and method are needed to protect the distributed networks of enterprises against unauthorized access to their valuable information assets by managing and enforcing the complex security policy requirements of the organization.

SUMMARY OF THE INVENTION

In accordance with the present invention, a system and method are disclosed to manage and enforce complex security requirements for a computer system in a distributed computer network.

It is therefore an object of the present invention to provide an access control system that can manage individual transactions by users around well-defined, detailed objects within an application. It is also an object of the present invention to provide a policy manager that enables the creation, modification, querying, and analysis of an enterprise access-control policy, as well as the configuration and monitoring of integrated audit logs, while delivering the performance and scalability required to meet the demands of any enterprise. It is a further object of the present invention to provide a system that combines a centrally managed policy database with distributed authorization (access control) services that enforce the policy for all applications across the organization.

It is also an object of this invention to provide a system that works in conjunction with any authentication system, including digital certificates and smartcards, and obviates the need for single sign-on systems by letting organizations set detailed, dynamic rules for exactly who can access which applications, databases, and other network objects. It is a still further object of this invention to provide a robust

security policy and authorization service that can be implemented in very heterogeneous environments, across all applications and databases within the organization, thereby completely eliminating the need for embedded, custom security code within applications, and making it

5 possible to centrally manage and administer a consistent, robust security policy for all applications, databases, and network resources.

Furthermore, organizations would no longer have to rely on authorization mechanisms provided by packaged or web application vendors that do not integrate with in-house or other third-party products.

10 In the preferred embodiment, the system comprises a policy manager located on a server for managing and distributing a local client policy based on a global security policy, and an application guard located on a client or server associated with one or more clients for managing access to securable components as specified by the local client policy.

15 The global policy specifies access privileges of the user to securable components. The policy manager may then distribute a local client policy based on the global policy to the client or server. An application guard located on the client or server then manages authorization requests to the securable components as specified by the local client

20 policy. Each authorization request may be recorded in an audit log to keep track of the authorization requests, whether they were granted or denied, and other useful information.

The system and method of the present invention supports centralized management and distributed authorization. A central policy server stores and manages the policy rules in a centrally administered database. A powerful graphical user interface is used to create, manage, and customize the elements of a policy. Security rules can be specified by both novice and expert users. A dedicated authorization service is associated with one or more applications. The central policy server automatically distributes (over the network) only the relevant portion of the enterprise policy to each remote service. This distributed architecture ensures that authorization requests are not bottlenecked at a central service point and provides unlimited scalability and maximum performance, regardless of the number of applications or policy rules involved.

A more sophisticated security policy is possible because the application has the ability to evaluate access privileges upon every access to the information, during every transaction, and at every data request.

Therefore, the present invention more efficiently and effectively manages and protects computer applications, databases, and network resources against unauthorized access in a distributed computer network.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of one embodiment for one system, in accordance with the present invention;

5

FIG. 2 is a block diagram of one embodiment of the non-volatile memory located within the server in FIG. 1, according to the present invention;

10

FIG. 3 is a block diagram of one embodiment of the non-volatile memory located within the client in FIG. 1, according to the present invention;

15

FIG. 4 is a block diagram of one embodiment of the policy manager located within the non-volatile memory in FIG. 2, in accordance with the present invention;

20

FIG. 5 is a block diagram of one embodiment of the application guard located within the non-volatile memory in FIG. 3, according to the present invention;

FIG. 6 is a block diagram of one embodiment of a policy loader, in accordance with the present invention;

FIG. 7 is a flowchart of one embodiment of method steps to configure a system, in accordance with the present invention;

5 FIG. 8 is a flowchart of one embodiment to manage policy in the management station, according to the present invention;

FIG. 9 is a flowchart of one embodiment to navigate tree in the management station, according to the present invention;

10 FIG. 10 is a flowchart of one embodiment to analyze policy in the management station, in accordance with the present invention;

15 FIG. 11 is a flowchart of one embodiment to edit policy in the management station, in accordance with the present invention;

FIG. 12 is a flowchart of method steps to distribute policy, according to one embodiment of the present invention;

20 FIG. 13 is a flowchart of method steps for client access authorization, in accordance with one embodiment of the present invention; and

FIG. 14 is a flowchart of method steps to evaluate authorization request, according to one embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention relates to an improvement in security techniques to protect computer systems against unauthorized access.

5 The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment will be readily apparent to those skilled in the art and the generic principles herein may be applied to other embodiments. Thus, the present invention is not intended to be limited to the
10 embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

The present invention includes a system and method for managing and enforcing complex security requirements in a distributed computer
15 network, and comprises a policy manager located on a server for managing and distributing a policy to a client, and an application guard located on the client, the application guard acting to grant or deny access to various components of the client, as specified by the policy.

Referring now to FIG. 1, a block diagram of one embodiment of a
20 distributed computer network system 110 is shown, including a server 112 connected via a network 114 to a client 116, in accordance with the present invention. One client 116 is shown, but server 112 is typically connected to many clients 116. In the FIG. 1 embodiment, server 112

may preferably include a central processing unit (CPU) 118, a read-only memory (ROM) 120, a random-access memory (RAM) 122, a non-volatile memory 124, an input device 126, and a display 128 all connected via a bus 130.

5 Similarly client 116 may preferably include a central processing unit (CPU) 132, a read-only memory (ROM) 134, a random-access memory (RAM) 136, a non-volatile memory 138, an input device 140, and a display 142 all connected via a bus 144.

10 Server 112 preferably contains a program stored in non-volatile memory 124 for managing a policy or a set of rules and then distributing the policy to client 116 via link 114. Client 116 preferably contains a program stored in non-volatile memory 138 for granting or denying access to various components or resources of client 116, as specified by the policy distributed from server 112. For example, various components
15 or resources of client 116 can include applications, functions or procedures within an application, data structures within an application, and database or file system objects referenced by an application.

Referring now to FIG. 2, a block diagram of one embodiment for non-volatile memory 124, located within server 112 of FIG. 1, is shown.

20 In the FIG. 2 embodiment, non-volatile memory 124 includes a policy manager 210 that manages and distributes a policy. A policy is intended to specify the security requirements for applications and database objects. A policy may contain thousands of "security rules" that describe

several constraints, including what applications a particular user can access, what objects (operations) within an application a user can access, and how those privileges are constrained by time, geography, or external events. In general, a policy or authorization policy should constrain
5 access to both applications and the operations within them. The policy may be generalized to groups and hierarchies, not just specified for individual users. This would greatly improve manageability and lead to more comprehensible, business-level policies.

An authorization policy preferably consists of four components,
10 including objects, subjects, privileges, and conditions. Objects may be applications, or the operations within an application. Examples of objects include applications or methods, web pages, database tables or files, and menu items in a graphical user interface. The granularity of objects has a direct impact on the level of security achieved. The less
15 information an object contains, it is less likely that a user has access to information not needed to perform his job function. On the other hand, the granularity of objects should be balanced against the ease of security management. The more information an object contains, the fewer objects that have to be protected, and the smaller the policy is.

20 Objects are preferably organized into an object hierarchy. If an object represents an application, then its children objects might represent the methods with the application. Similarly, if an object

represents a database, then its children objects might represent the tables and views within the database.

If a user is granted a certain privilege on a parent object, then he is automatically granted the privilege on all the children objects. Similarly, 5 if a user is denied a certain privilege on a parent object, then he is automatically denied the privilege on all the children objects. In other words, privileges are inherited from parent to children objects. Privilege inheritance through the object hierarchy eases security management because rather than granting the same privilege to every child object, the 10 privilege is granted once to the parent object, and if the privileges of an object change, the policy on all the children objects automatically reflect the changes made to the object.

Subjects may be users, or roles containing users, who access protected objects. Subjects correspond to users that have access to 15 information in a system. Users can either be internal or external to a system. Users are authorized to access information in order to perform their job functions. Such access may be controlled so that a user gets access only to the information needed to perform his job function.

An object, such as an application or a database, typically has its 20 own list of users. These are users who can log on to the object and be authenticated by the objects, sometimes through an external authentication server. In a large system, users are preferably maintained separately by one or more directory servers. Users are preferably

extracted from objects or directory servers, and are maintained up-to-date by synchronizing with these objects and directory servers.

Alias users may also be supported. An alias of a user is another user who inherits all the privileges of the user under certain conditions.

5 Alias facilitates authorization management by providing fine granularity of control on the propagation of privileges. For example, an alias of a user can be created to perform his job function while he is absent. The inheritance of privileges takes effect only when the user is absent. An alias implements the business requirements of delegation, where the
10 privileges of a user can be delegated to another user under certain conditions. Conditional inheritance of privileges through an alias reduces the burden of security management, because it restricts privilege propagation to situations when certain conditions are satisfied.

Users of an object may be defined as being local to that object. In
15 a typical system, the same user is often represented by different login identifications in different objects. This system may support the notion of a "global" user to capture this situation. Every global user is mapped to a set of local users, one per object. Global users facilitate the centralized management of users throughout the system, even if they are
20 identified by different names in different objects.

A privilege defines the kinds of access that may be allowed on objects. In the preferred embodiment, a privilege is the right to perform a particular action on a specific object. The kinds of privileges that apply

to an object depend on the type of the object. Examples of privileges include the right to execute an application, the right to download a web page, the right to query a database table, or the right to view a menu item.

5 Privileges are granted to users so they can accomplish tasks required for their job. A privilege should be granted to a user only when it is absolutely required for the user to accomplish a task. Excessive granting of unnecessary privileges may lead to compromised security. A user may receive a privilege in two different ways, privileges can be
10 granted to users explicitly (for example, user SMITH can be granted the privilege to execute the payroll application), or privileges can be granted to a role (a named group of privileges), which is then granted to one or more users (for example, a role named "clerk" can be granted the privilege to execute the payroll application, and user SMITH can be
15 granted the clerk role).

Roles are named groups of privileges that are granted to users or other roles. Users granted to a role are the members of that role. A role is often used to represent the set of privileges needed to perform a job function.

20 The members of a role automatically inherit all the privileges granted or denied to the role. In addition, roles may be organized into a role hierarchy, where parent roles are granted to children roles. If a parent role is granted a privilege, then the children roles are

automatically granted the privilege. Similarly, if a role is denied a privilege, then the children roles are automatically denied the privilege.

Roles of an object may be defined as being local to that object. In a typical system, the same role is often represented by different names in different objects. This system may support the notion of a “global” role to capture this situation. Every global role is mapped to a set of local roles, one per object. Global roles facilitate the centralized management of roles throughout the system, even if they are identified by different names in different objects.

Role membership may be further constrained by the notion of mutual exclusion. Two roles are mutually exclusive if no single user can be granted to both roles simultaneously. Role mutual exclusion implements a business requirement of separation of duty. For example, a submit_budget role and an approve_budget role should be mutually exclusive, because no user should be simultaneously authorized to perform both actions.

In a typical policy, there are preferably two types of access rules, a grant rule, and a deny rule. A grant rule states that a privilege on an object is granted to a subject under an optional constraint. A deny rule states that a privilege on an object is denied to a subject under an optional constraint. Additionally, a wild card “any” may be used as a privilege, object, or subject, meaning that any legitimate value could be substituted in its place.

An access request preferably consists of a privilege, an object, and a subject, representing the fact that the subject request authorization of the privilege on the object. An access request matches a grant rule if the privilege, object, and subject match those in the rule, and the constraint in the rule evaluates to "true." An access request matches a deny rule if the privilege, object, and subject match those in the rule, and the constraint in the rule does not evaluate to "false."

An access request is denied if there is a deny rule matching the request, or there are no access rules matching the request. An access request is granted if there are no deny rules matching the request, and there is a grant rule matching the request.

Conditions define the constraints on when objects and subjects can be accessed. The constraints in an access rule specifies further requirements on when the access rule is applicable. These requirements could be conditioned on properties of the object or the subject.

Constraints are preferably expressions formed from conditions and Boolean operators NOT, AND, and OR. Three kinds of built-in conditions may be used: 1) relational operations =, <>, <, <=, >, >= on integers; 2) relational operations =, <>, LIKE, NOTLIKE on strings (the operator LIKE takes a string and a pattern and evaluates to true if the string matches the pattern, the operator NOTLIKE is the negation of LIKE); and 3) set operations IN, NOTIN (the operator IN on integers takes an integer and a set of integers and evaluates to "true" if the integer is in the set, the

operator IN on strings is similarly defined, and the operator NOTIN is the negation of IN).

In addition to built-in conditions, users of system 110 may declare custom evaluation functions, which are customer-defined conditions.

5 System 110 may provide an Application Programming Interface (API) for invoking customer-supplied code to evaluate custom evaluation functions. For example, an evaluation function could access a remote database to validate certain properties of the object. Another evaluation function could invoke an external server to authenticate the subject.

10 Now referring to the FIG. 2 embodiment, policy manager 210 preferably includes a management station program 212 to operate policy manager 210, a distributor program 214 to distribute local client policies to clients, a logger program 216 to track authorization requests, and a database management system (DBMS) 218 to maintain policy data files.

15 Policy manager 210 also includes an audit log data file 220 to record authorization requests, an optimized policy data file 222, an enterprise policy data file 224, an administrative policy data file 226, and a local administrative policy data file 228. The contents and operation of policy manager 210 are further discussed below in conjunction with FIGS. 4, 8,

20 9, 10, 11, and 12.

Referring now to FIG. 3, a block diagram of one embodiment for non-volatile memory 138, located within client 116 of FIG. 1, is shown. In the FIG. 3 embodiment, non-volatile memory 138 preferably includes

an application guard 310 that grants or denies access to various components of client 116, as specified by a pre-determined policy. For example, various components of client 116 can include applications, data, and/or objects. In the FIG. 3 embodiment, application guard 310 preferably includes at least one application 312, an authorization library program 314, an authorization engine program 316, and a local client policy 318. The contents and operation of application guard 310 are further discussed below in conjunction with FIGS. 5, 13, and 14.

Referring now to FIG. 4, a block diagram of one embodiment for policy manager 210, located within non-volatile memory 124 in FIG. 2, is shown. In the preferred embodiment, policy manager 210 allows system users to implement, analyze, edit and update a centrally-managed security policy or enterprise policy 224. In the FIG. 4 embodiment, policy manager 210 preferably includes a management console or management station 212, a database management system 218, an audit facility or logger 216, and a distributor 214.

In the FIG. 4 embodiment, management station 212 preferably includes a graphical user interface (GUI) 410 for creating or customizing rules by system users. Management station 212 supports concurrent rule development by multiple users. Each policy rule preferably includes four basic components: 1) an object that is to be protected; 2) an access right or privilege; 3) a global or local user to which the privilege applies; and 4) conditions under which the privilege is granted or denied,

including built-in access criteria such as time of day or location, as well as custom-defined access criteria.

Graphical user interface (GUI) 410 provides a user-friendly set of menu options or management services 412 to fully operate the policy manager. Programs controlled by the menu options may include navigation 414, search 416, distribution 418, edit 420, query 422, and log viewer 424. The operation of these programs are further discussed below in conjunction with FIGS. 8, 9, 10, 11, and 12. As an alternative to the GUI, the management services can be operated from an application through an API that allows programs to perform the same functions as a human operator. In the preferred embodiment, management station also includes an application guard 426 to allow only authorized administrators to operate management station 212. Local administrative policy 228 provides a set of policy rules specifying which users are authorized to access management station 212.

After the policy rules are created or modified using management station 212, they may then be distributed to appropriate clients 116. Management station 212 includes a communication interface 434 in order to pass information between various other components in system 110.

Prior to when the policy rules are distributed, a parser/type checker 428 preferably reviews and reconstructs the policy rules to make sure that they are syntactically and semantically correct according to a

predefined policy language. The policy rules pass through a database layer (DB layer) 430 and an open database connectivity layer (ODBC) 432 before being stored as enterprise policy 224. DB layer 430 formats the policy rules into standard database storage tables, and ODBC 432 provides a common interface to various vendor-specific databases.

Enterprise policy 224 is then passed to distributor 214. An optimizer program 436 within distributor 214 determines which application guard 310 needs to receive which policy rules. A differ program 438 determines what type of changes were made to optimized policy 222, and then distributes only the changed policy rules or local client policy 318 to the appropriate application guards 310 through an ODBC layer 440 and a communication interface 442, which enforce access control to local applications and data.

Since the application guards 310 can be distributed among various clients 116, and each application guard 310 has its own specific local client policy 318, the system provides scalability.

Distributor 214 may also be used to optimize administrative policy 226 into an optimized administrative policy or local administrative policy 228 for use with application guard 426 in management station 212.

Referring now to FIG. 5, a block diagram of one embodiment of application guard 310, located within non-volatile memory 138 in FIG. 3, is shown. Application guard 310 may be distributed on clients 116

throughout an enterprise, and is designed to reside along with each of the protected applications having an associated application guard 310.

Application guard 310 supports transactional access control by allowing an application to be aware of the authorization service and to make authorization requests at each and every user interaction, data request, or business-level transaction. In addition, the design and integration of application guard 310 is fundamental to providing access control to business-level objects within an application since the authorization services have visibility to those named policy objects within the application.

In the FIG. 5 embodiment, application guard 310 is preferably integrated into application 312 through a high-level application programming interface (API) or authorization library 314 that allows application 312 to request authorization services as needed through an application guard interface 512. Typically, this can be done very quickly by including authorization requests at key points in application 312 for control user interaction or database access so that each interaction is protected with a minimum of development.

In the FIG. 5 embodiment, an authorization request is processed by authorization engine 316. A parser/type checker 514 parses local client policy 318 and stores the parsed local client policy in RAM 136. An evaluator 516 then determines whether the authorization request should be granted or denied by evaluating the authorization request with

the parsed local client policy in RAM 136. Plug-ins 522 in authorization engine 316 allow for additional capabilities to process and evaluate authorization requests based on customized code. Each authorization request is then recorded in an audit log 518 and transmitted to logger 216 via a communication interface 520.

Users have the option of implementing application guard 310 locally to application 312, as a service running on the same system as application 312, or as a remote authorization service through a remote procedure call to another server. The advantage of the latter design would be to offload the application server from handling authorization services or allowing a single authorization server to handle a multiple number of applications. A local implementation would provide maximum performance and minimize any network traffic overhead.

As seen in FIG. 5, application guard 310 includes an application guard interface 512 coupled to an application 312 for requesting access to securable components. Application guard 310 also includes at least one authorization engine 316 for evaluating requests from application guard interface 512 as specified by local client policy 318. Multiple authorization engines 316 can be used for added performance and reliability. Furthermore, application guard interface 512 can be located on a client computer, while authorization engine 316 and local client policy 318 can be located on a client server.

The application guard authorization service of the present invention introduces virtually no performance overhead to an existing application 312. The policy rules developed at policy manager 210 are compiled into an optimized form before being distributed to the target application guards 310. This optimized form only distributes attributes relevant to that application guard 310, so that access requests may be evaluated by reviewing only a few rules rather than frequently analyzing the potentially large policy rule base.

Referring back to FIG. 4, logger 216 may then advantageously receive a client audit log 450 through a communication interface 452 and an ODBC 454 from authorization engine 316 (FIG. 5). Client audit log 450 is then formatted by message processing 456 before being stored in audit log 220. Audit log 220 may then be monitored via log viewer 424 in management station 212.

Referring now to FIG. 6, a block diagram of one embodiment of a policy loader 610 is shown. In the FIG. 6 embodiment, policy rules may be entered one at a time into enterprise policy database 224 via management station 212 (FIG. 4), or the policy rules may alternatively be loaded as a batch process via policy loader 610. Policy loader 610 is an additional utility that bulk loads an existing set of policy rules into enterprise policy database 224. An existing set of policy rules may be entered into policy loader 610 via input 612. A parser/type checker 614 then preferably reviews and reconstructs the policy rules to make sure

that they are syntactically and semantically correct according to a predefined policy language. The policy rules may then be passed through a DB layer 616 and an ODBC 618 before being stored in enterprise policy database 224.

5 Referring now to FIG. 7, a flowchart of method steps to configure a system in accordance with one embodiment of the present invention is shown. Initially, in step 710, a system administrator installs policy manager 210 on a server 112. The installation may include management station 212, distributor 214, logger 216, and DBMS 218. After all the
10 components of policy manager 210 have been installed, the system administrator then enters a set of policy rules. In step 712, the administrator can decide whether to use policy loader 610, or to use management station 212 to enter policy rules. If the system administrator decides to use management station 212, then at step 714,
15 policy rules are entered using edit function 420. However, if policy loader 610 is used, then at step 716, policy rules are entered into a file, and at step 718, the file of policy rules passes to policy loader 610.

Next, in step 720, the system administrator installs application guards 310 onto client systems 116, as well as installing local client
20 policies 318 onto client systems 116. Then at step 722, the system administrator registers plug-ins 522 into application guards 318 to allow for additional capabilities in order to process authorization requests based on customized code.

Referring now to FIG. 8, a flowchart of one embodiment to manage policy under management services 412 in management station 212 is shown. In order to allow for a complex set of policy rules, a number of different functionality features are incorporated into system 110 to
5 facilitate implementation and management.

In the FIG. 8 embodiment, at step 810, an authorized administrator logs in to policy manager 210. Next, in step 812, the authorized administrator chooses between administrative mode or enterprise mode. The administrative mode allows the system
10 administrator to manage administrative policy 226, and the enterprise mode allows the system administrator to manage enterprise policy 224. The system administrator is then presented with six menu options including navigate tree 814, analyze policy 816, edit policy 818, distribute policy 820, view audit log 822, and exit 824. The features of
15 navigate tree 814, analyze policy 816, edit policy 818, and distribute policy 820 are described in more detail through FIGS. 9, 10, 11, and 12, respectively. View audit log 822 is a security feature that allows an administrator to view and track all authorization requests that have occurred at any application guards 310 connected to system 110. The
20 system administrator can choose any of the menu options, or at step 824 the system administrator may then exit the system.

Referring now to FIG. 9, a flowchart of one embodiment of menu option navigate tree 814 in management station 212 is shown. Navigate

tree 814 provides a set of options for an administrator to add, delete, and/or modify features on server 112 or client 116. The features that an administrator may add, delete, and/or modify include global users 910, global roles 912, directories 914, local roles 916, local users 918, applications 920, application guards 922, and declarations 924. At step 926, the system administrator may then exit from navigate tree 814.

Referring now to FIG. 10, a flowchart of one embodiment of menu option analyze policy 816 in management station 212 is shown. Analyze policy 816 preferably allows an authorized user to analyze and view rules and policies within enterprise policy 224. At step 1010, the user has an option to search rules, or at step 1012 to query policy. When search rules is selected, a search can be made for all the grant rules or all the deny rules pertaining to a particular user. When query policy is selected, a search can be made on who is granted or denied what privilege on which objects under what conditions.

After analyzing and viewing rules or policies, at step 1014 the system administrator may exit from analyze policy 816.

Referring now to FIG. 11, a flowchart of one embodiment of menu option edit policy 818 in management station 212 is shown. Edit policy 818 allows an authorized user to add, delete, and/or modify enterprise policy 224 features. The features that may be edited include rule sets 1110, access 1112, privilege 1114, objects 1116, user/role 1118, and

attributes 1120. At step 1122, the system administrator may then exit edit policy 818.

Referring now to FIG. 12, a flowchart of one embodiment of method steps of menu option distribute policy 820 is shown. After enterprise policy 224 has been initially entered or modified in any way, the modified features of enterprise policy 224 may then be distributed to appropriate application guards 310. At step 1210, upon selecting the distribute policy option, distributor 214 optimizes enterprise policy 224. Then at step 1212, differ 438 preferably computes any difference between the newly optimized policy and optimized policy 222. At step 1214, the newly optimized policy is then published as optimized policy 222 in DBMS 218. Next, at step 1216, only the changed portions of optimized policy 222 are committed to appropriate application guards 310. At step 1218, application guards 310 receive the changed policy, and then at step 1220, application guards 310 merge the changed policy into local client policy 318. Next at step 1222, new local client policy 318 is activated to work with application guard 310.

Referring now to FIG. 13, a flowchart of one embodiment of method steps for client access authorization is shown. The FIG. 13 example for using a standard application guard 310 by a user begins with a user requesting access to a securable component protected by an application guard 310. In step 1310, application guard 310 constructs and issues an authorization request. At step 1312, the authorization request is

evaluated by application guard 310 according to its local client policy 318 to determine whether to allow or deny the authorization request. At step 1314, audit 518 then records the authorization request in audit log 450. Next, at step 1316, if there is an error in the authorization request, or if the request is not valid, then at step 1318 the user is denied access. However, if the authorization request is valid, then at step 1320 it is determined whether access should be granted. If the evaluated authorization request does not deny access for the user, then at step 1322 access is allowed. If the evaluated authorization request denies access for the user, then at step 1324 access is denied.

Referring now to FIG. 14, a flowchart of one embodiment of method steps to evaluate an authorization request from an application guard 310 is shown. In order to evaluate an authorization request at application guard 310, in step 1420, evaluator 516 first searches any deny rules in local policy 318. At step 1412, if evaluator 516 finds any deny rules, then at step 1414, an evaluation is performed on any constraints on the deny rules. If, at step 1416, the evaluation finds presently valid constraints on the deny rules, then at step 1418 access is denied. However, if at step 1416, the evaluation finds that the constraints on the deny rules are not presently valid, or if no deny rules are found in foregoing step 1412, then at step 1420, a search of grant rules is performed. If no grant rules are found at step 1422 that would allow access for the user, then at step 1418, access is denied. If in step 1422

grant rules are found, then at step 1424 an evaluation is performed on any constraints in the grant rules. If the evaluated constraint is presently valid, then at step 1426, a true value is passed, and at step 1428 access is allowed. However, if the evaluated constraint is not presently valid, then at step 1426, a false value is passed, and at step 1418 access is denied.

The invention has been explained above with reference to a preferred embodiment. Other embodiments will be apparent to those skilled in the art in light of this disclosure. For example, the present invention may readily be implemented using configurations other than those described in the preferred embodiment above. Additionally, the present invention may effectively be used in conjunction with systems other than the one described above as the preferred embodiment. Therefore, these and other variations upon the preferred embodiments are intended to be covered by the present invention, which is limited only by the appended claims.

WHAT IS CLAIMED IS:

1 1. A system for maintaining security in a distributed computing
2 environment, comprising:
3 a policy manager for managing a security policy; and
4 an application guard for managing access to securable components
5 as specified by the security policy.

1 2. The system of claim 1, wherein said policy manager comprises a
2 management station for constructing and editing the security policy.

1 3. The system of claim 2, wherein said policy manager further
2 comprises a distributor for distributing the security policy to a client.

1 4. The system of claim 2, wherein said policy manager further
2 comprises a distributor for distributing a customized local policy based
3 on the security policy to a client.

1 5. The system of claim 4, wherein said policy manager further
2 comprises a logger for recording and tracking authorization events that
3 occur through the application guard.

1 6. The system of claim 4, wherein the policy manager further
2 comprises a database management system for maintaining the security
3 policy.

1 7. The system of claim 4, wherein the customized local policy is
2 optimized.

1 8. The system of claim 1, wherein said securable components are
2 selected from the group consisting of: at least one application, a function
3 within an application, a procedure within an application, a data
4 structure within an application, a database object referenced by an
5 application, or a file system object referenced by an application.

1 9. The system of claim 1, wherein said system is scalable by further
2 comprising a plurality of clients, said policy manager further managing
3 and distributing a customized local policy to each client, and at least one
4 additional application guard located on each client for managing access
5 to the securable components as specified by each customized local
6 policy.

1 10. The system of claim 1, wherein said application guard includes an
2 application guard interface coupled to an application for requesting
3 access to the securable components, and at least one authorization
4 engine for evaluating requests from the application guard interface as
5 specified by a customized local policy based on the security policy.

1 11. The system of claim 10, wherein said application guard interface is
2 located on a client, and said at least one authorization engine and said
3 customized local policy are located on a client server.

1 12. The system of claim 1, wherein the security policy is defined by a
2 policy language to grant or deny access to the securable components for
3 a particular user.

1 13. The system of claim 1 further comprising a policy loader for bulk
2 loading the security policy onto the system.

1 14. The system of claim 1, wherein said policy manager includes a set
2 of menu options to manage and distribute the security policy.

1 15. The system of claim 14, wherein said set of menu options include:
2 navigate tree, analyze policy, edit policy, distribute policy, and view audit
3 log.

1 16. The system of claim 1, wherein the application guard further
2 allows for additional customized code to process and evaluate
3 authorization requests based on the additional customized code.

1 17. The system of claim 1, wherein the policy manager further includes
2 a policy manager application guard for managing access to the policy
3 manager as specified by a local administrative policy.

1 18. A system for controlling user access in a distributed computing
2 environment, comprising:
3 a global policy specifying access privileges of the user to securable
4 components;
5 a policy manager located on a server for managing and distributing
6 a local client policy based on the global policy to a client, and
7 an application guard located on the client for managing access to
8 the securable components as specified by the local client
9 policy.

1 19. The system of claim 18 further comprising at least one additional
2 client, said policy manager further managing and distributing a
3 customized local policy based on the global policy to each additional
4 client, and at least one additional application guard located on each
5 additional client for managing access to the securable components as
6 specified by the customized local policy.

1 20. The system of claim 18, wherein said policy manager comprises:
2 a management station for constructing and editing the global
3 policy; and
4 a distributor for distributing the local policy to the client.

1 21. The system of claim 20, wherein said policy manager further
2 comprises a database management system for maintaining the global
3 policy.

1 22. The system of claim 18, wherein said policy manager further
2 comprises a logger for recording and tracking authorization events that
3 occur through the application guard.

1 23. The system of claim 18, wherein said securable components are
2 selected from the group consisting of: at least one application, a function
3 within an application, a procedure within an application, a data
4 structure within an application, a database object referenced by an
5 application, or a file system object referenced by an application.

1 24. The system of claim 18, wherein the global policy is defined by a
2 policy language to grant or deny access to the securable components for
3 a particular user.

1 25. The system of claim 18, wherein said system is scalable by further
2 comprising a plurality of clients, said policy manager further managing
3 and distributing a customized local policy to each client, and at least one
4 additional application guard located on each client for managing access
5 to the securable components as specified by each customized local
6 policy.

1 26. The system of claim 18, wherein said application guard includes
2 an application guard interface coupled to an application for requesting
3 access to the securable components, and at least one authorization
4 engine for evaluating requests from the application guard interface as
5 specified by the local client policy.

1 27. The system of claim 18 further comprising a policy bulk loader for
2 bulk loading the global policy onto the system.

1 28. The system of claim 18, wherein the local client policy is optimized.

1 29. The system of claim 18, wherein the application guard further
2 allows for additional customized code to process and evaluate
3 authorization requests based on the additional customized code.

1 30. The system of claim 18, wherein the policy manager further
2 includes a policy manager application guard for managing access to the
3 policy manager as specified by a local administrative policy.

1 31. A system for authorization that provides access to securable
2 components for a user, comprising:
3 a policy specifying access privileges of the user to the securable
4 components;
5 an application guard; and
6 a processor coupled to said system, said processor executing said
7 application guard to manage access to the securable
8 components.

1 32. The system of claim 31 further including an audit log for recording
2 and tracking each authorization event that occurs though the application
3 guard.

1 33. The system of claim 31, wherein said securable components are
2 selected from the group consisting of: at least one application, a function
3 within an application, a procedure within an application, a data
4 structure within an application, a database object referenced by an
5 application, or a file system object referenced by an application.

1 34. The system of claim 31, wherein said application guard comprises
2 an application guard interface connected to an application for managing
3 access to the securable components in said application.

1 35. The system of claim 31, wherein the application guard further
2 allows for additional customized code to process and evaluate
3 authorization requests based on the additional customized code.

1 36. The system of claim 31, wherein said application guard includes
2 an application guard interface coupled to an application for requesting
3 access to the securable components, and at least one authorization
4 engine for evaluating requests from the application guard interface as
5 specified by a customized local policy based on the policy.

1 37. The system of claim 36, wherein said application guard interface is
2 located on a client, and said at least one authorization engine and said
3 customized local policy are located on a client server.

1 38. A system for managing security in a distributed computing
2 environment, comprising:
3 a policy manager; and
4 a processor coupled to said system, said processor executing said
5 policy manager to manage and distribute a customized local
6 policy based on a global policy to a client.

1 39. The system of claim 38, wherein said policy manager comprises a
2 management station for constructing and editing the global policy.

1 40. The system of claim 39, wherein said policy manager further
2 comprises a distributor for distributing the customized local policy to the
3 client.

1 41. The system of claim 38, wherein said policy manager further
2 comprises a logger for recording and tracking authorization events that
3 are received from the client.

1 42. The system of claim 38, wherein said global policy specifies access
2 privileges of at least one user to securable components.

1 43. The system of claim 38, wherein the policy manager comprises a
2 database management system for maintaining the global policy.

1 44. The system of claim 38, wherein the global policy is defined by a
2 policy language to grant or deny access to securable components for a
3 particular user.

1 45. The system of claim 38, wherein said policy manager includes a set
2 of menu options to manage and distribute the customized local policy.

1 46. The system of claim 38, wherein the customized local policy is
2 optimized.

1 47. The system of claim 38, wherein the policy manager further
2 includes a policy manager application guard for managing access to the
3 policy manager as specified by a local administrative policy.

1 48. A method for maintaining security in a distributed computing
2 environment, comprising the steps of:
3 managing a policy using a policy manager by specifying access
4 privileges of a user to securable components; and
5 distributing the policy to a client having an application guard,
6 whereby the application guard manages access to the
7 securable components as specified by the policy.

1 49. The method of claim 48, further including the step of recording
2 authorization events that occur through the application guard after
3 distributing the policy.

1 50. The method of claim 48, wherein the securable components are
2 selected from the group consisting of: at least one application, a function
3 within an application, a procedure within an application, a data
4 structure within an application, a database object referenced by an
5 application, or a file system object referenced by an application.

1 51. A method for maintaining security on a client in a distributed
2 computing environment, comprising the steps of:
3 constructing and issuing an authorization request for a user to
4 access to securable components located on the client using
5 an application guard;
6 evaluating the authorization request using the application guard to
7 determine if the authorization request is valid or invalid; and
8 allowing access to the user via the application guard if the
9 evaluated authorization request was valid, and denying
10 access to the user via the application guard if the
11 authorization request was invalid.

1 52. The method of claim 51, after evaluating the authorization request,
2 further including the step of recording the authorization request in an
3 audit log.

1 53. A computer-readable medium comprising program instructions for
2 maintaining security in a distributed computing environment by
3 performing the steps of:

4 managing a policy using a policy manager by specifying access
5 privileges of a user to securable components;
6 distributing the policy using the policy manager to a client having
7 an application guard, whereby the application guard
8 manages access to the securable components as specified by
9 the policy; and
10 executing said policy manager with a processor to manage and
11 distribute the policy.

1 54. A computer-readable medium comprising program instructions for
2 maintaining security on a client in a distributed computing environment
3 by performing the steps of:

4 constructing and issuing an authorization request for a user to
5 access to securable components located on the client using
6 an application guard;
7 evaluating the authorization request using the application guard to
8 determine if the authorization request is valid or invalid;
9 allowing access to the user via the application guard if the
10 evaluated authorization request was valid, and denying
11 access to the user via the application guard if the
12 authorization request was invalid; and
13 executing said application guard with a processor to automatically
14 maintain security on the client.

1 55. A system for maintaining security in a distributed computing

2 environment, comprising:

3 means for managing a policy using a policy manager by specifying

4 access privileges of a user to securable components;

5 means for distributing the policy using the policy manager to a

6 client having an application guard, whereby the application

7 guard manages access to the securable components as

8 specified by the policy; and

9 means for executing the policy manager to manage and distribute

10 the policy.

1 56. A system for maintaining security on a client in a distributed
2 computing environment, comprising:
3 means for constructing and issuing an authorization request for a
4 user to access to securable components located on the client
5 using an application guard;
6 means for evaluating the authorization request using the
7 application guard to determine if the authorization request is
8 valid or invalid;
9 means for allowing access to the user via the application guard if
10 the evaluated authorization request was valid, and denying
11 access to the user via the application guard if the
12 authorization request was invalid; and
13 means for executing said application guard to automatically
14 maintain security on the client.

SYSTEM AND METHOD FOR MAINTAINING SECURITY
IN A DISTRIBUTED COMPUTER NETWORK

ABSTRACT OF THE DISCLOSURE

5

A system and method for maintaining security in a distributed computing environment comprises a policy manager located on a server for managing and distributing a security policy, and an application guard located on a client for managing access to securable components as specified by the security policy. In the preferred embodiment, a global policy specifies access privileges of the user to securable components. The policy manager may then preferably distribute a local client policy based on the global policy to the client. An application guard located on the client then manages access to the securable components as specified by the local policy.

10

15

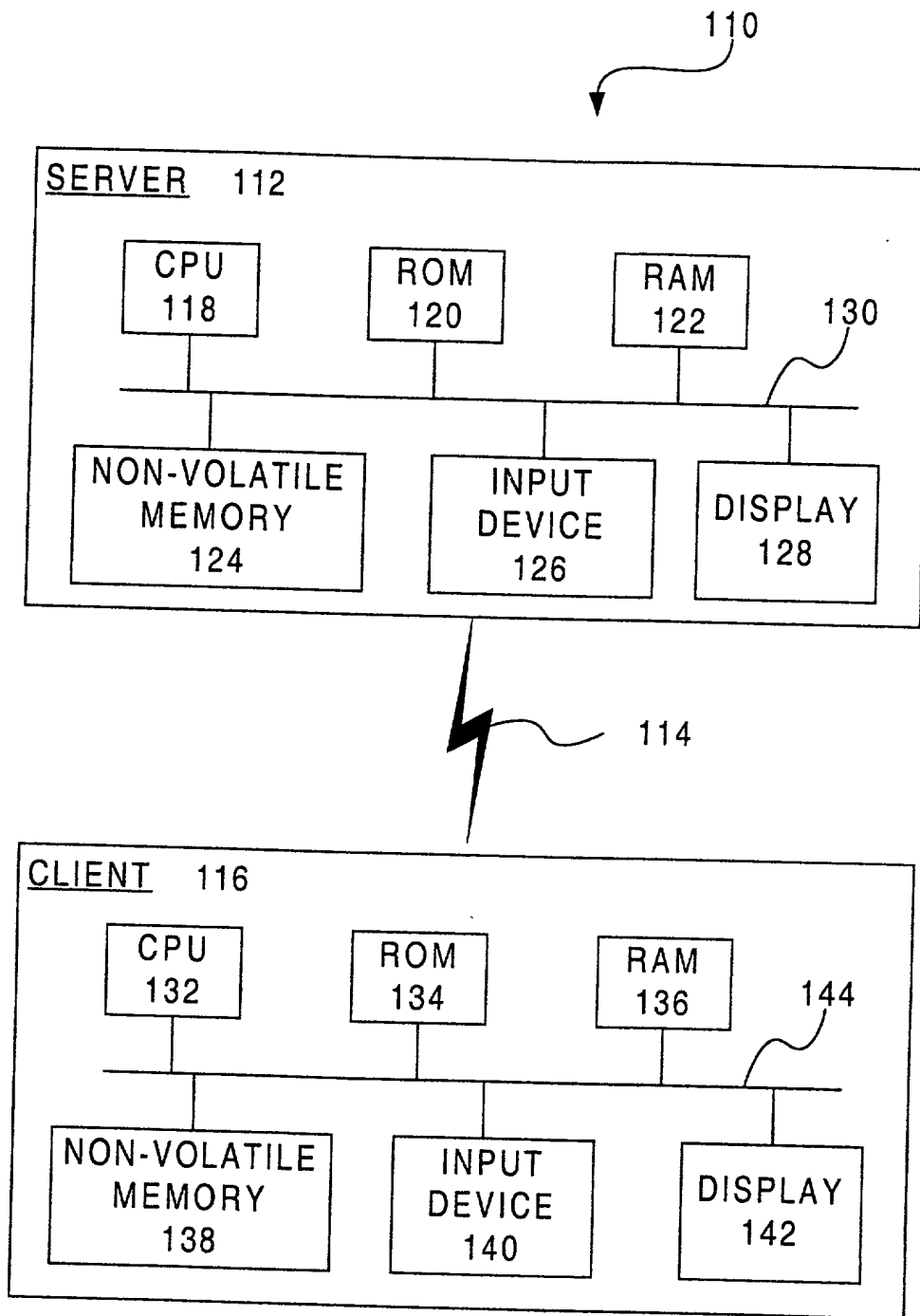


FIG. 1

SERVER

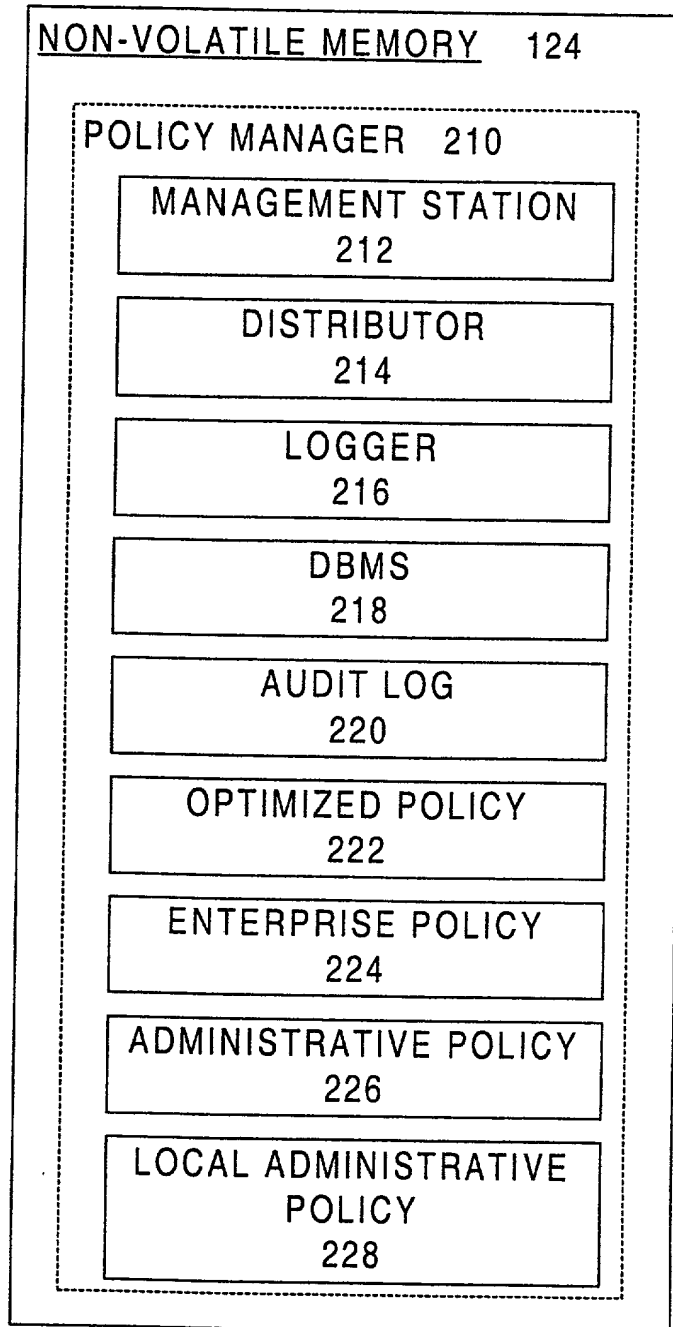


FIG. 2

CLIENT

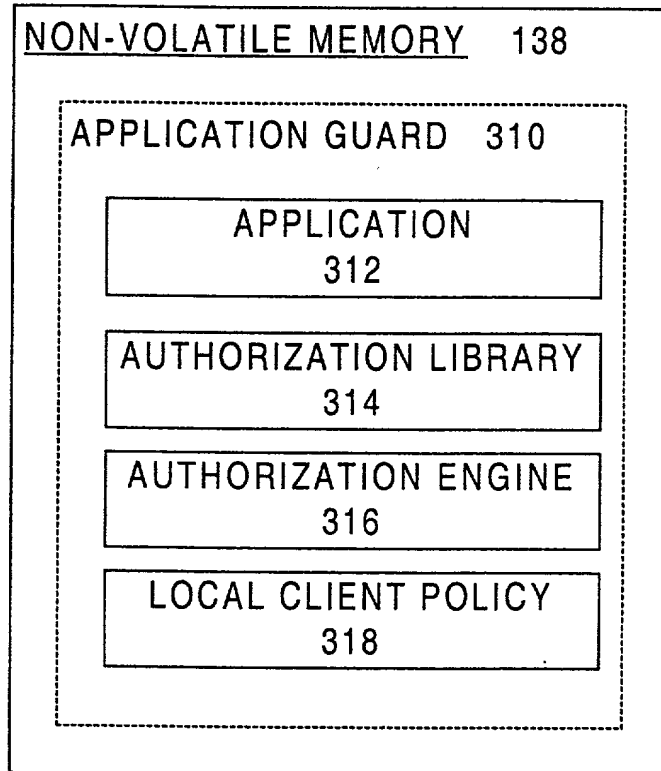


FIG. 3

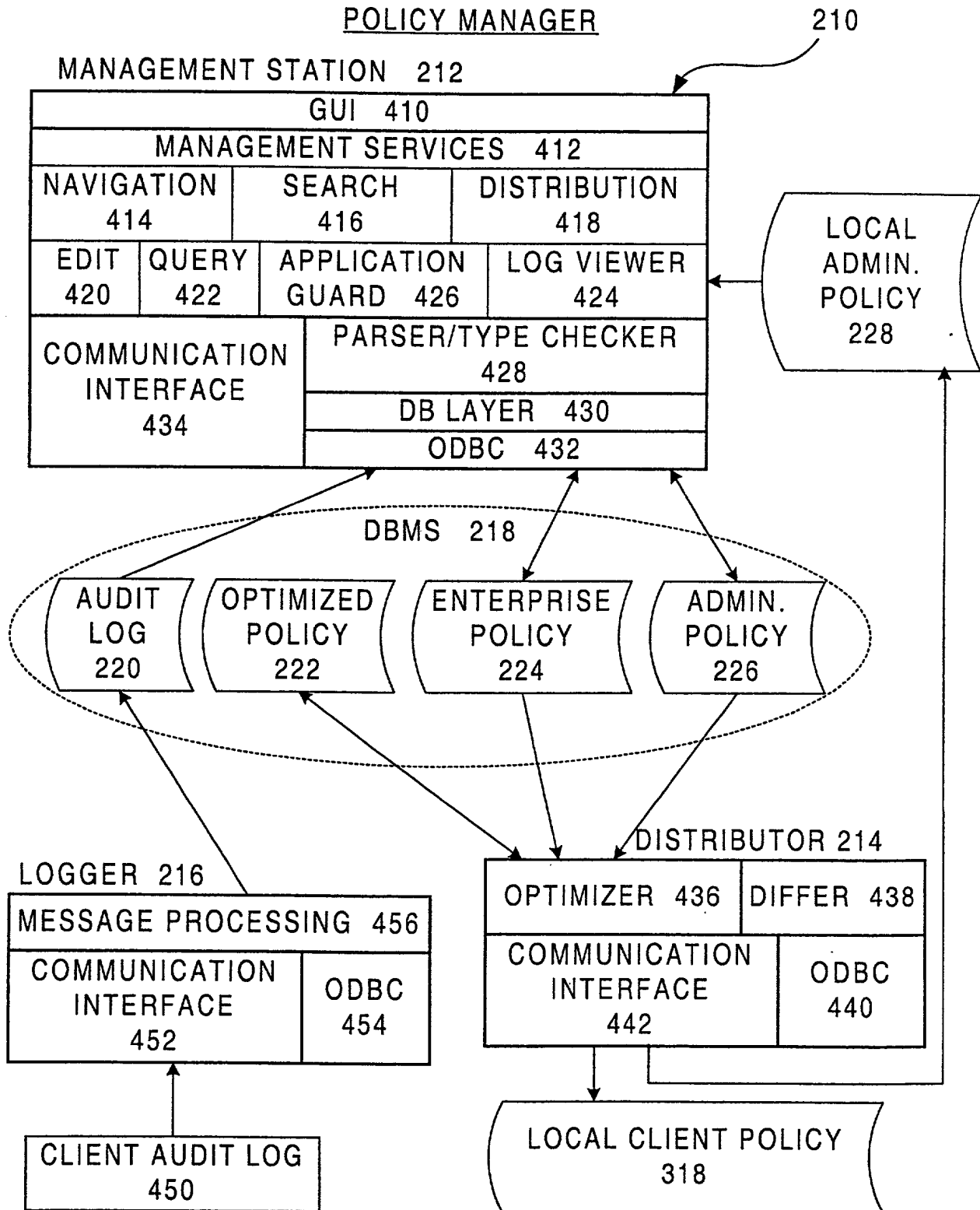


FIG. 4

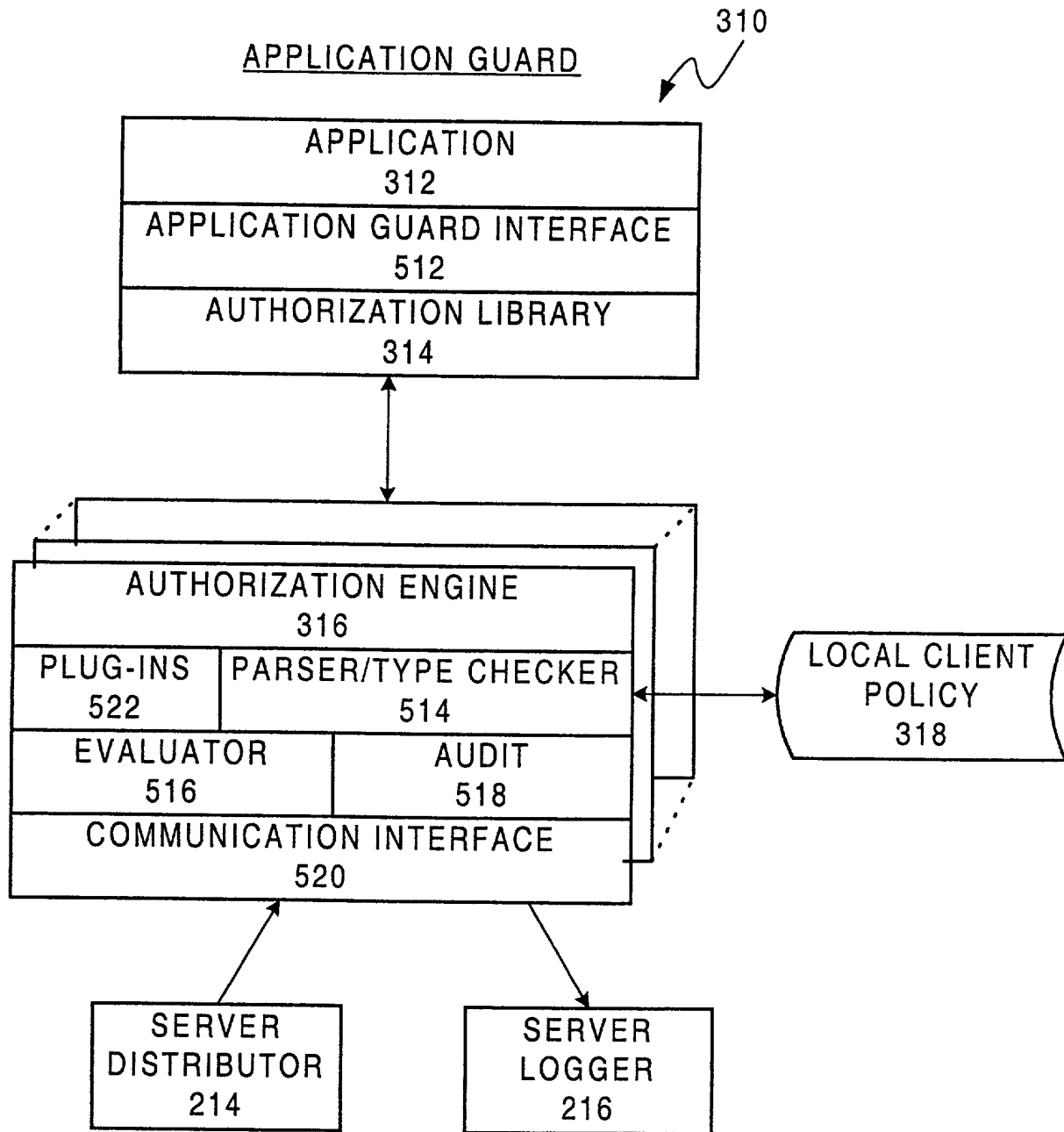


FIG. 5

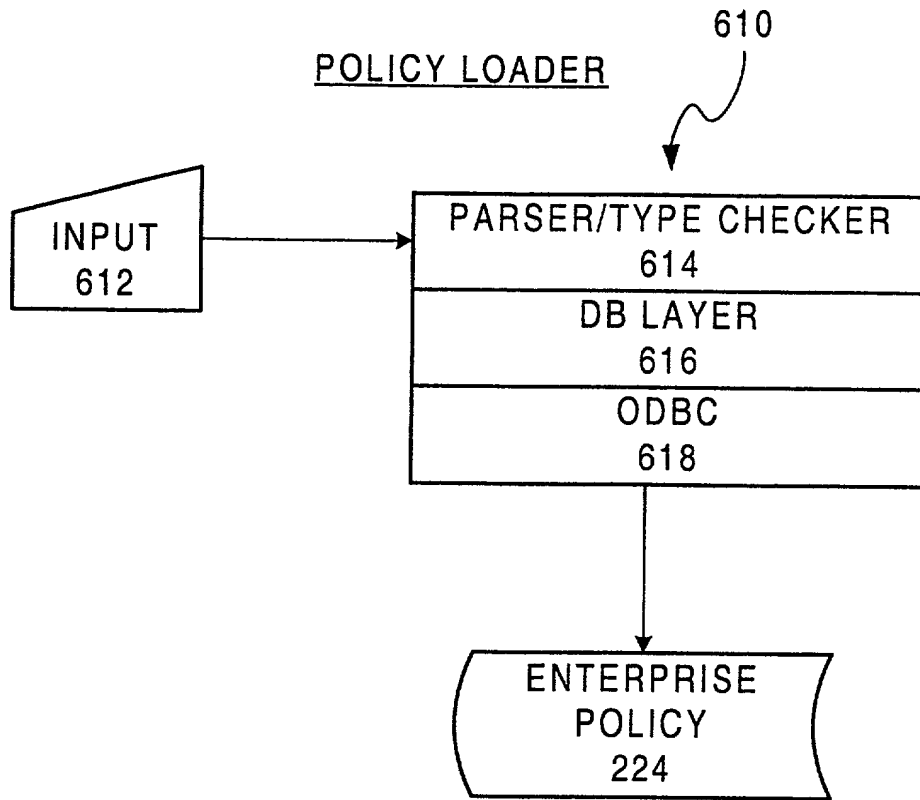


FIG. 6

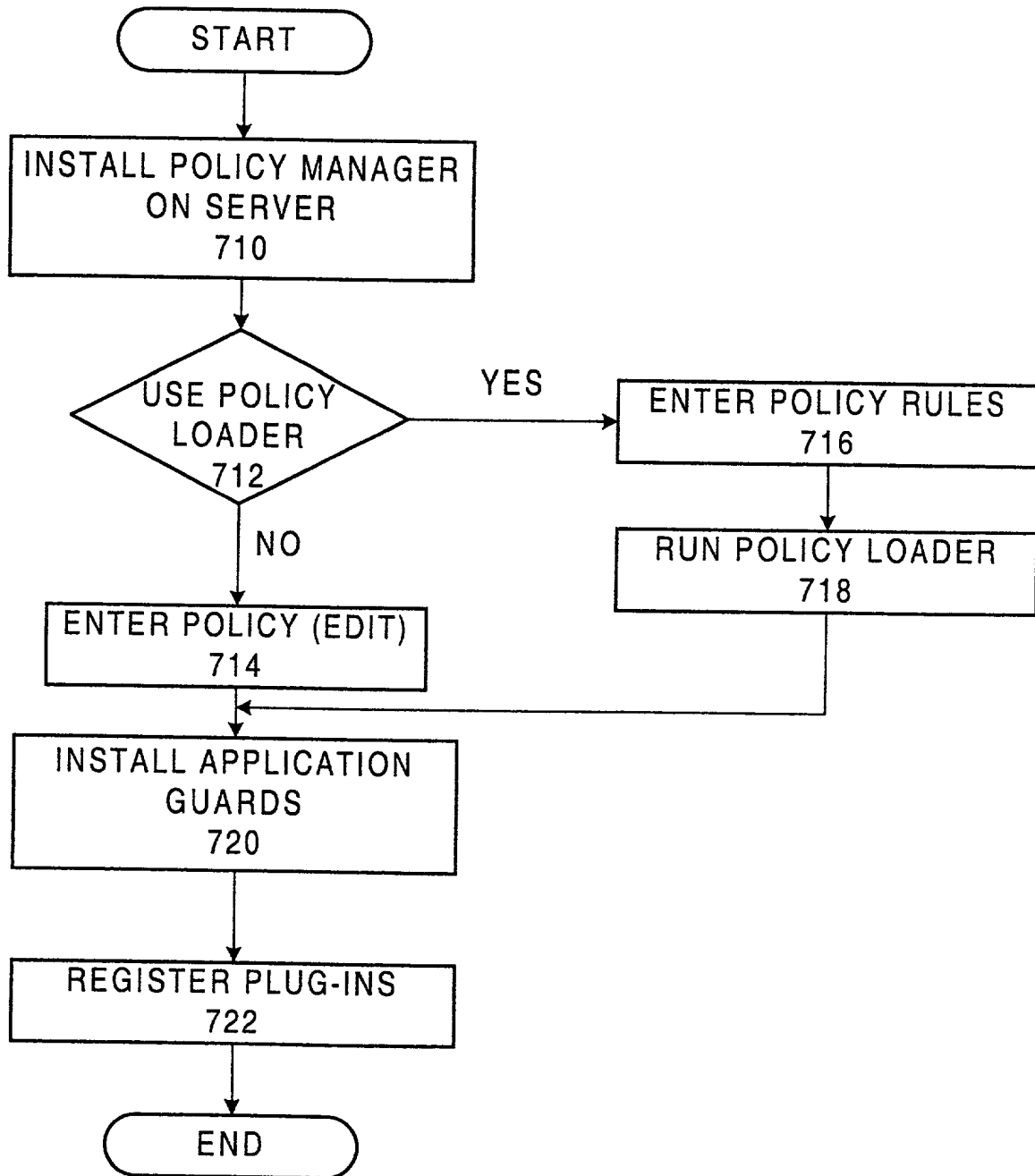
CONFIGURE SYSTEM

FIG. 7

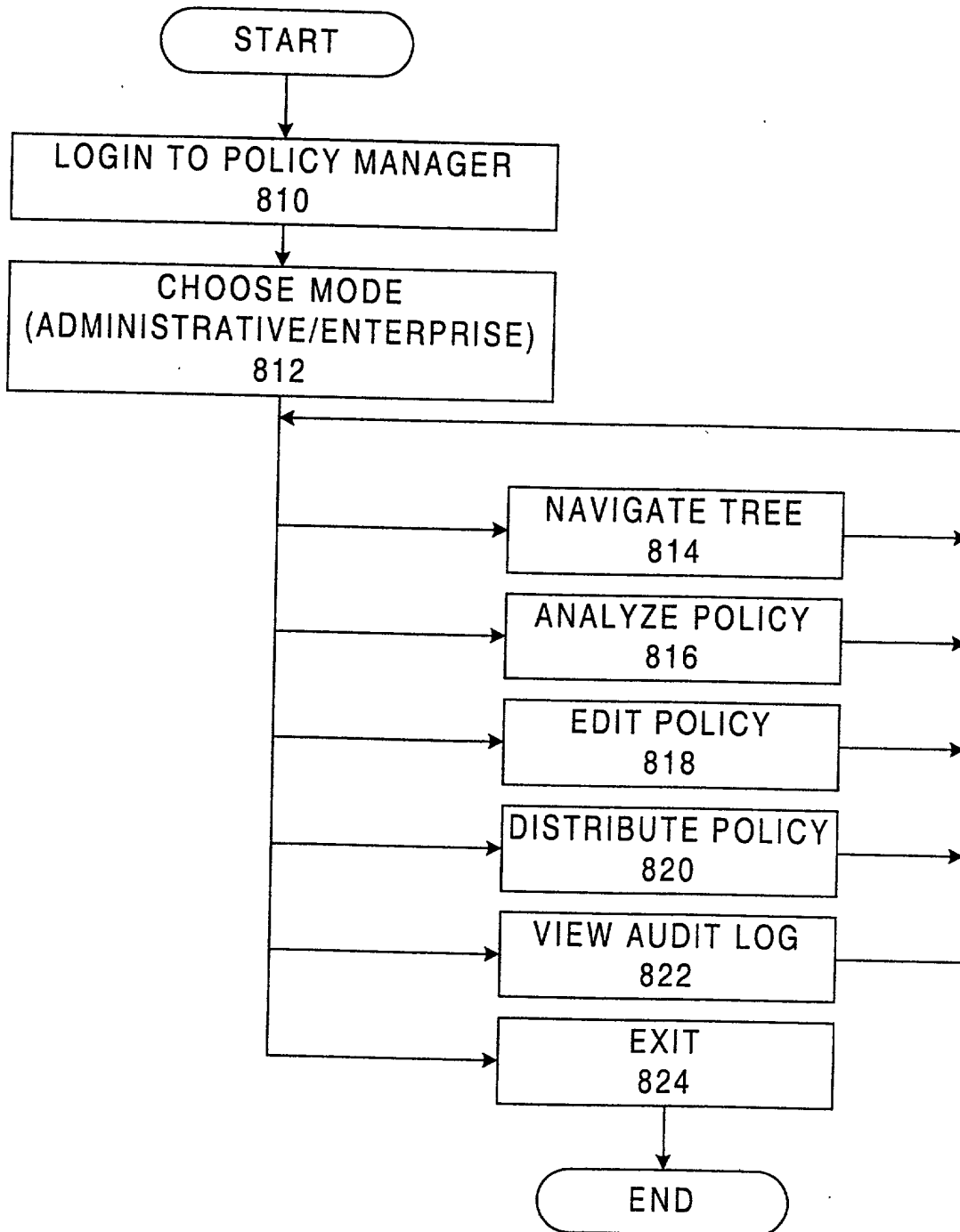
MANAGE POLICY

FIG. 8

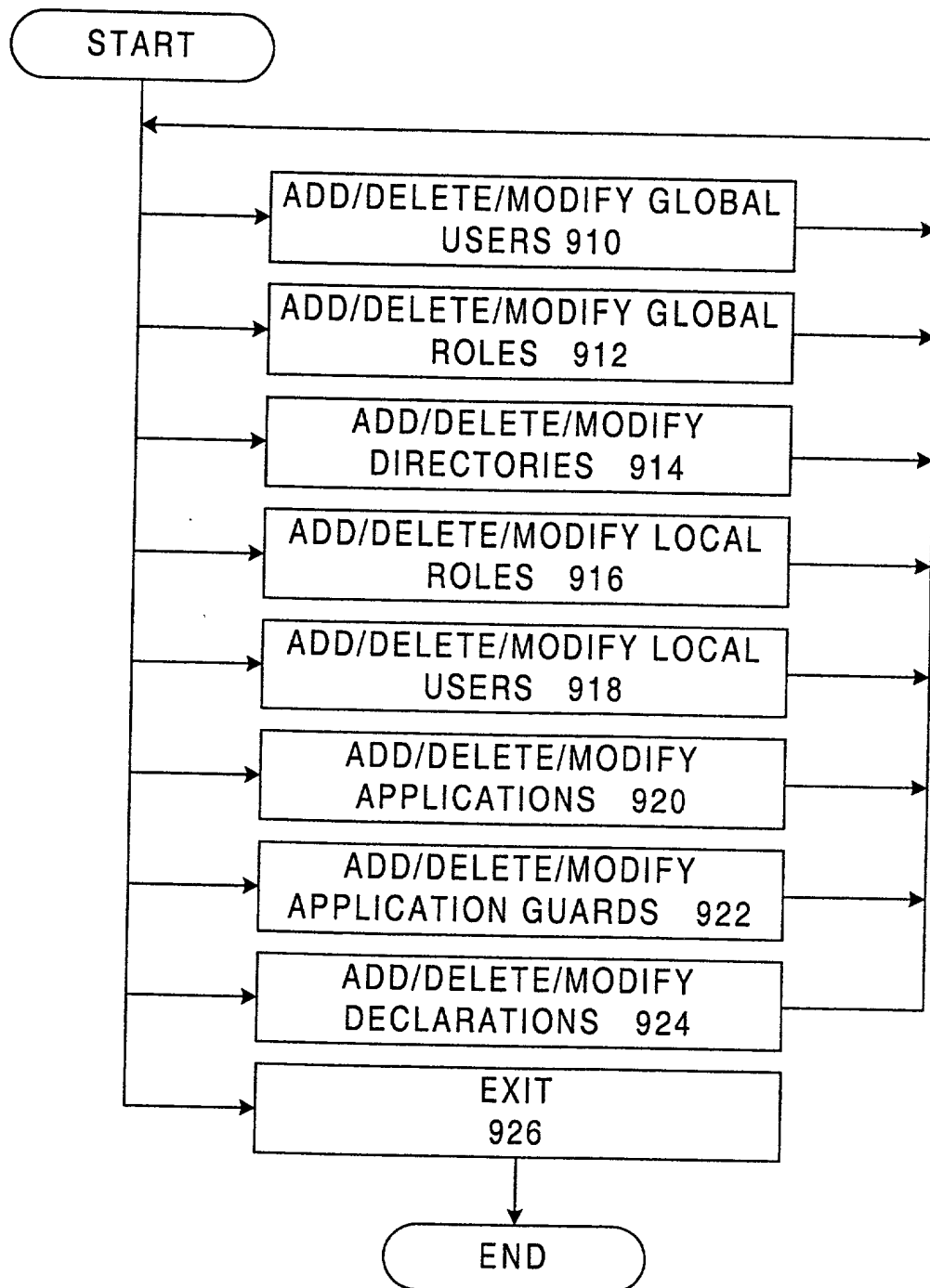
NAVIGATE TREE

FIG. 9

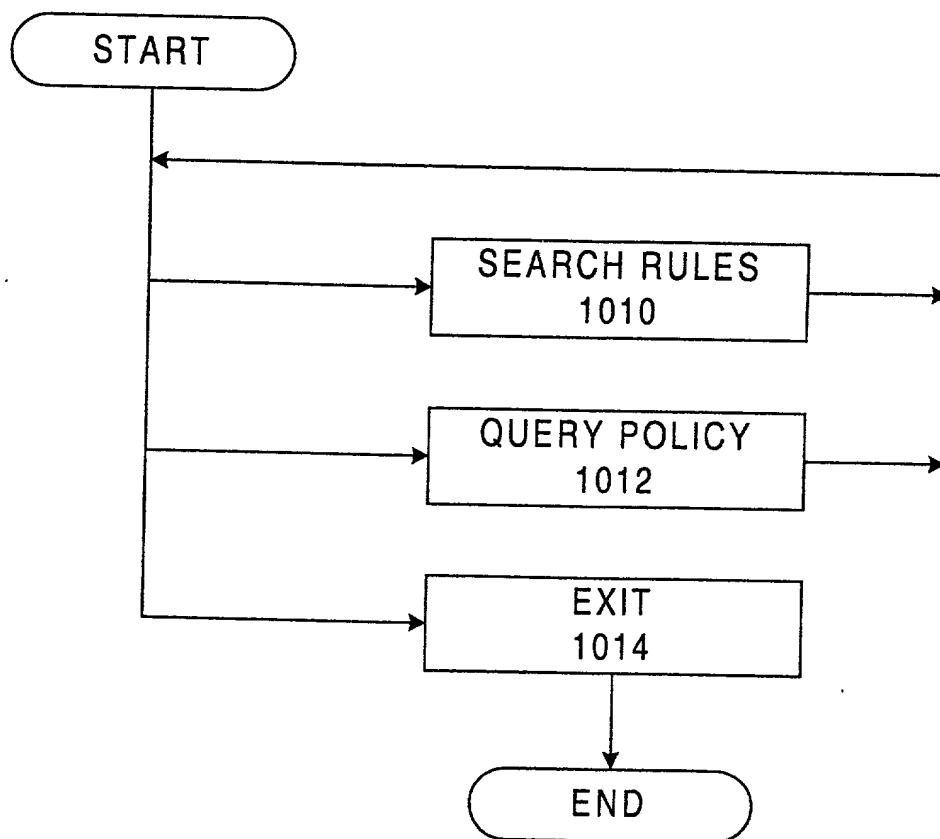
ANALYZE POLICY

FIG. 10

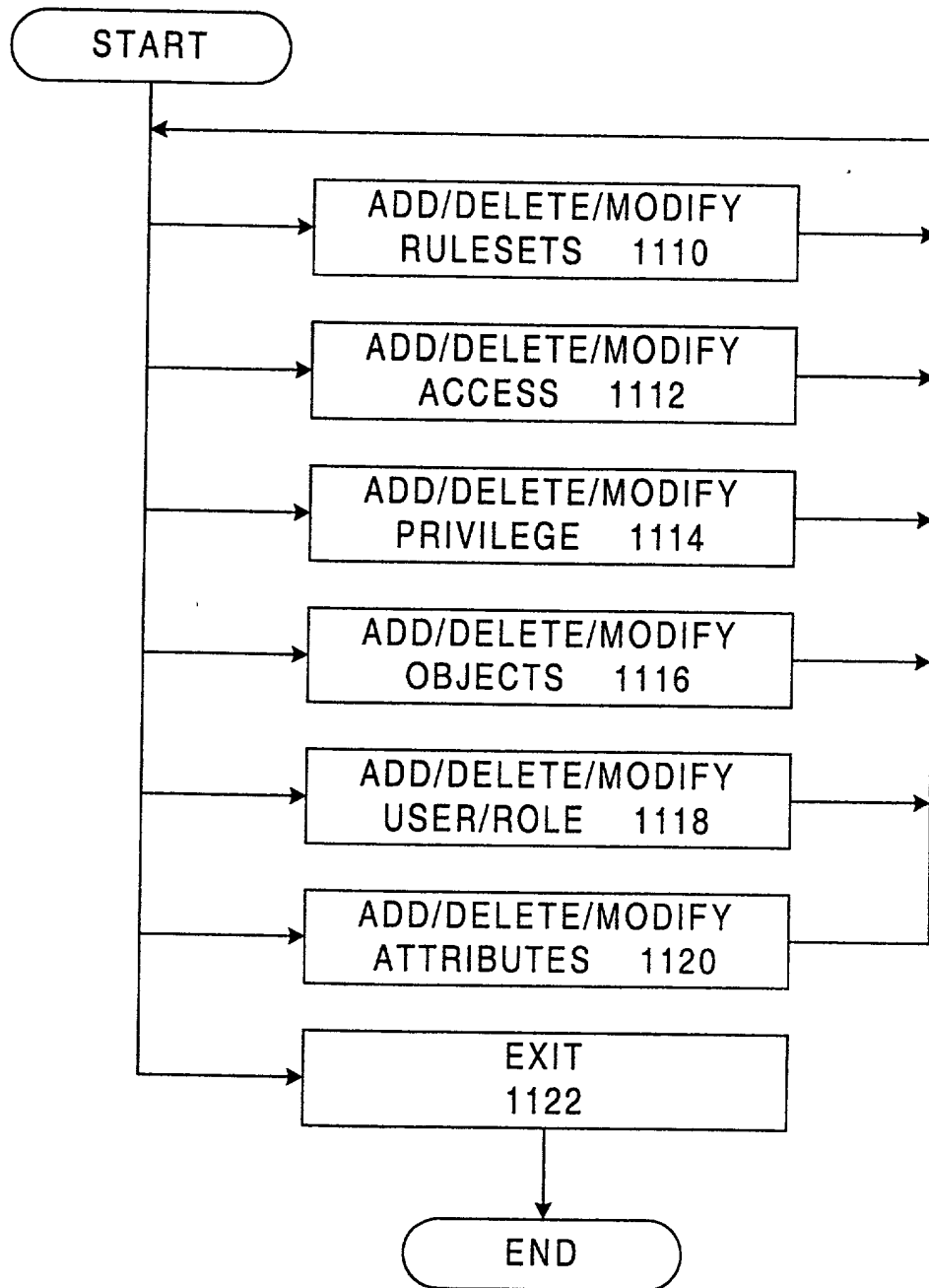
EDIT POLICY

FIG. 11

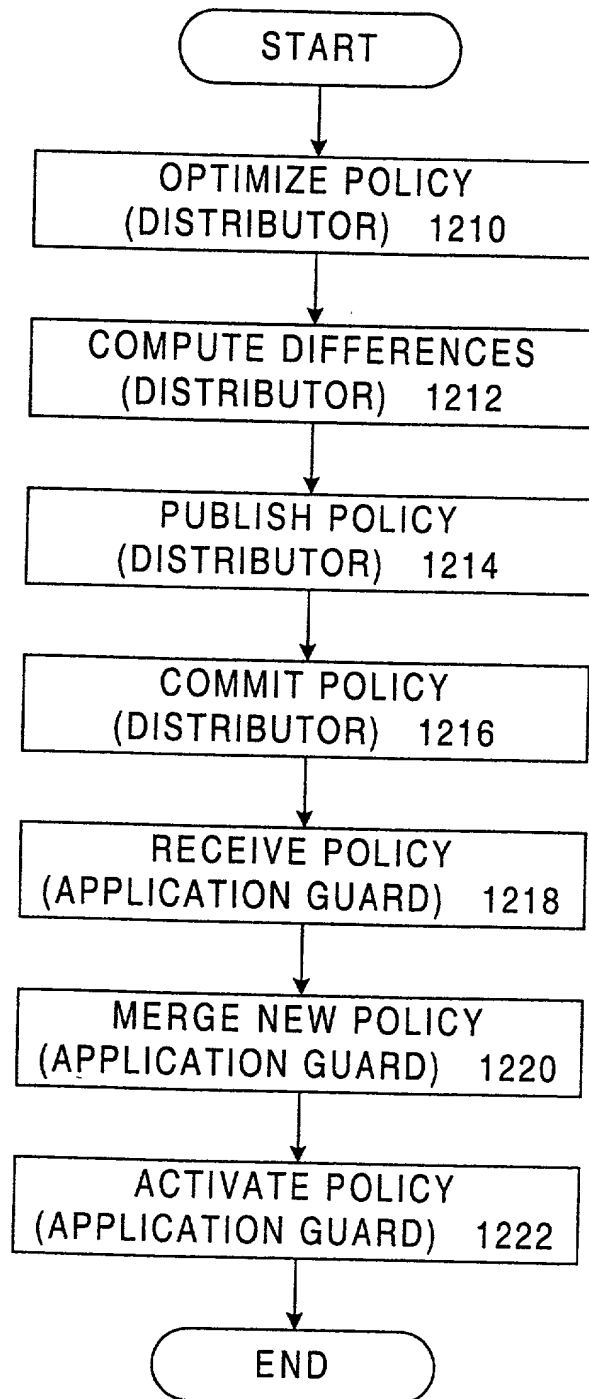
DISTRIBUTE POLICY

FIG. 12

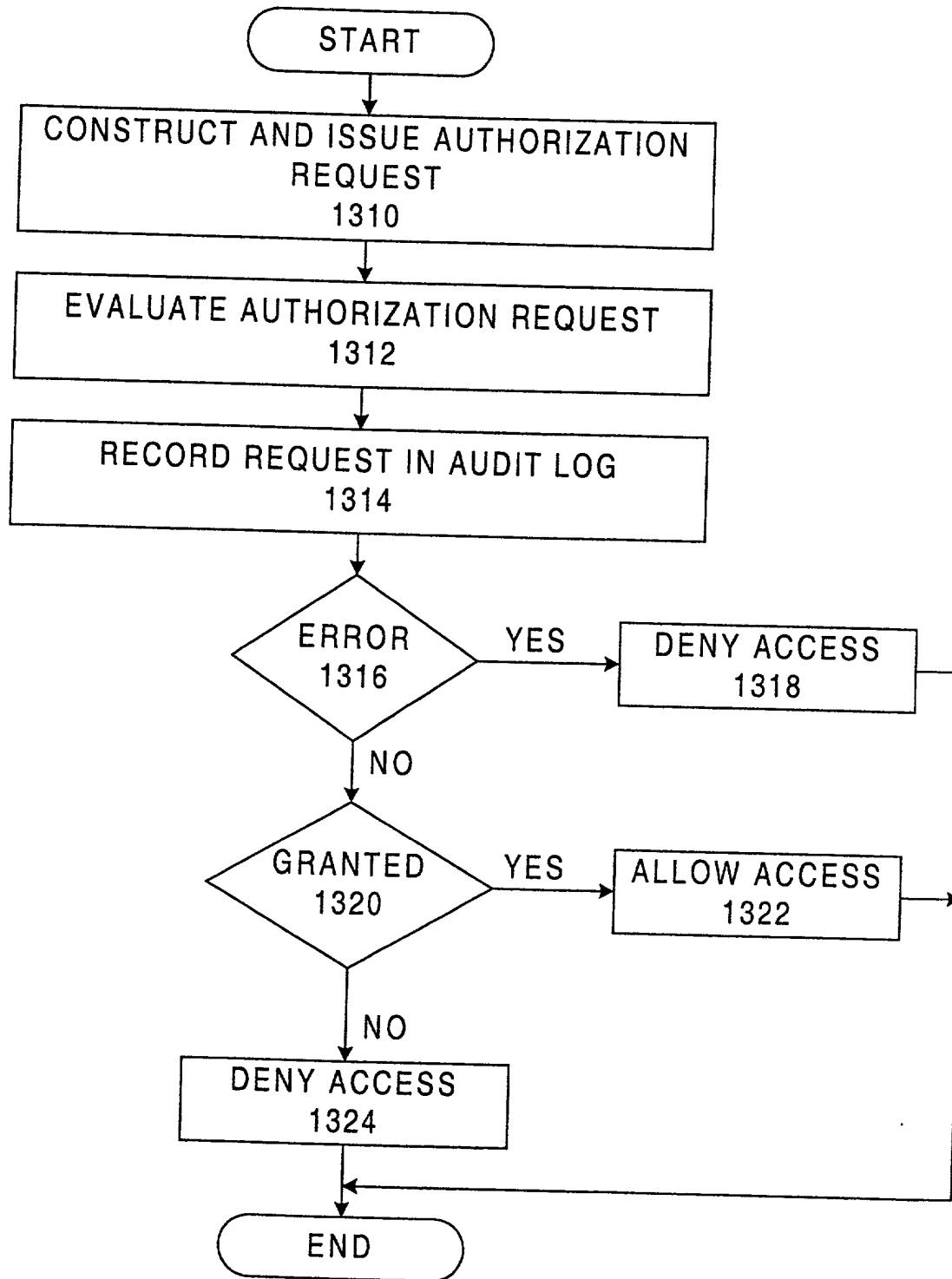
CLIENT ACCESS AUTHORIZATION

FIG. 13

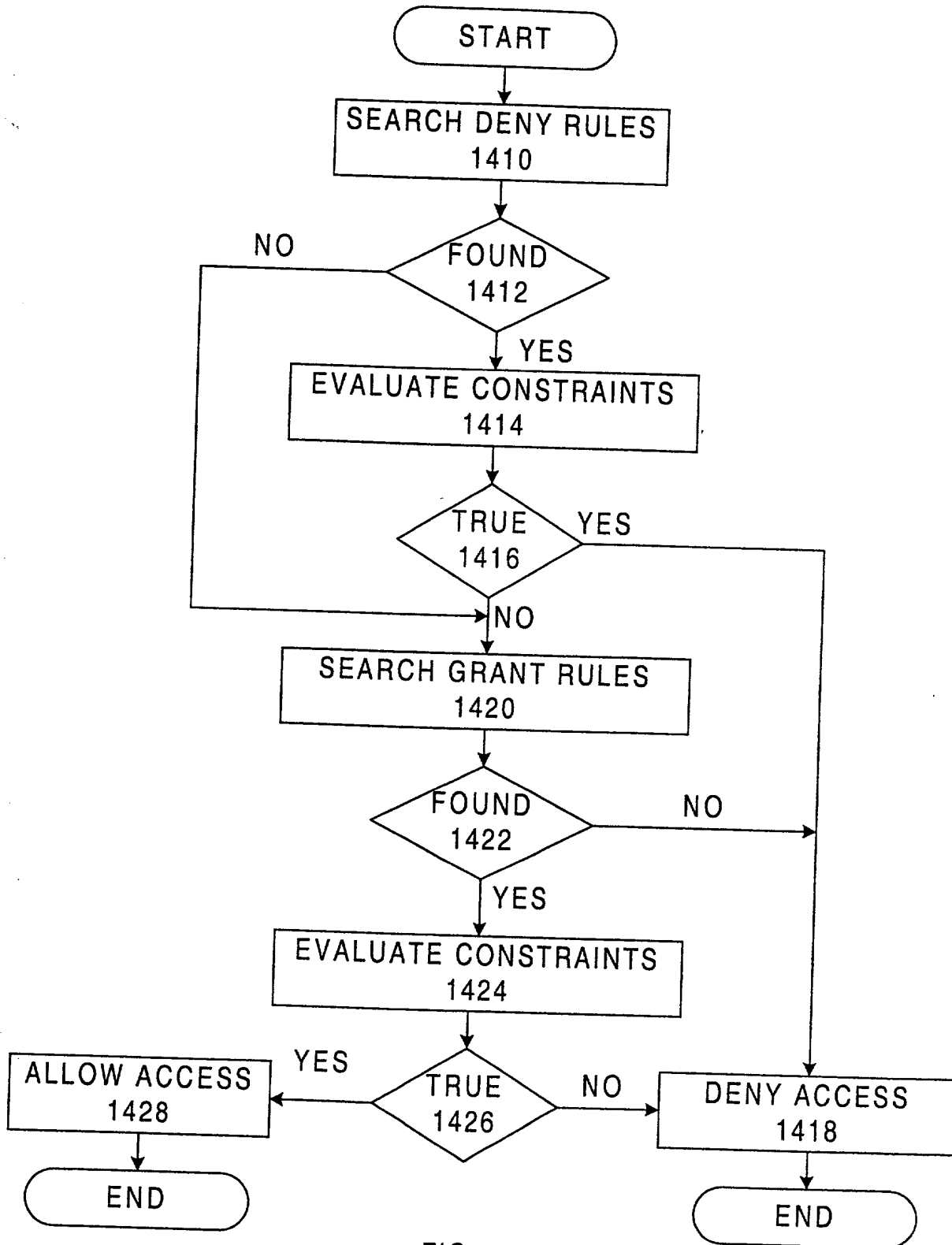
EVALUATE AUTHORIZATION REQUEST

FIG. 14

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am an original, first and joint inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled "**SYSTEM AND METHOD FOR MAINTAINING SECURITY IN A DISTRIBUTED COMPUTER NETWORK**," the specification of which (check one):

☒ is attached hereto.

☐ was filed on _____
as U.S. Application No. _____
or PCT International Application No. _____
and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment specifically referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, §1.56.

I hereby claim foreign priority benefits under Title 35, United States Code §119(a)-(d) or §365(b) of any foreign application(s) for patent or inventor's certificate, or §365(a) of any PCT International application which designated at least one country other than the United States, listed below and have also identified below any foreign application for patent or inventor's certificate, or PCT International application, having a filing date before that of the application on which priority is claimed.

Prior Foreign Application(s)

Priority Claimed

| | | | | |
|-------------------|--------------------|---------------------------------|------------------------------|-----------------------------|
| _____ (Number) | _____ (Country) | _____ (Day/Month/Year filed) | <input type="checkbox"/> Yes | <input type="checkbox"/> No |
| _____ (Number) | _____ (Country) | _____ (Day/Month/Year filed) | <input type="checkbox"/> Yes | <input type="checkbox"/> No |

I hereby claim the benefit under Title 35, United States Code §119(e) of any United States provisional application(s) listed below.

60/105963
(Application Number)

10/28/98
(Filing Date)

(Application Number)

(Filing Date)

I hereby claim the benefit under Title 35, United States Code §120 of any United States application(s), or §365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of Title 35, United States Code §112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, §1.56 which became available between the filing date of the prior application and the national or PCT International filing date of this application.

(Application Number)

(Filing Date)

(Status -- patented, pending, abandoned)

(Application Number)

(Filing Date)

(Status -- patented, pending, abandoned)

POWER OF ATTORNEY: I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith:

John S. Ferrell, Reg. No. 34,593; J. Eppa Hite, Reg. No. 30,266; John D. Henkhaus, Reg. No. 42,656; Charles B. Katz, Reg. No. 36,564; Gregory J. Koerner, Reg. No. 38,519; Dennis A. Nicholls, Reg. No. 42,036; Wendi R. Schepler, Reg. No. 43,091; Robert Toczycki, Reg. No. 38,341; Robert D. Wasson, Reg. No. 40,218; Franklyn Weiss, Reg. No. 23,041; Susan L. Yee, Reg. No. 41,388.

SEND ALL CORRESPONDENCE TO:

Robert Toczycki
CARR & FERRELL, LLP
2225 East Bayshore Road, Suite 200
Palo Alto, CA 94303
TEL: (650) 812-3400
FAX: (650) 812-3444

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor: Mark Moriconi

Inventor's signature [Signature] Dated: 2/9/99

Residence 11 Michaels Way, Atherton, CA 94027

Post Office Address 11 Michaels Way, Atherton, CA 94027 Citizenship U.S.A.

Full name of second joint inventor, if any: Shelly Qian

Second Inventor's signature [Signature] Dated: 2/9/99

Residence 20643 Greenleaf Drive, Cupertino, CA 95014

Post Office Address 20643 Greenleaf Drive, Cupertino, CA 95014 Citizenship U.S.A.